Technische Universiteit Eindhoven

Eindhoven University of Technology
address: Den Dolech 2
P.O. Box 513
5600 MB Eindhoven The Netherlands
Telephone: +31 (0) 40 247 31 10
Telefax: +31 (0) 40 246 41 51

Faculteit Technische Natuurkunde
Werkverband Transportfysica

Titel:

Experimental setup for measurements in turbulent pipe flow : hardware and software

Auteur: A.F. Moene

Verslagnummer: R-1467-P

Datum: July 1999

Contents

1 Introduction **4**

2 Hardware **6**
2.1 Flow system . . . 6
2.1.1 Generation of flow . . . 6
2.1.2 Pipe sections . . . 6
2.1.3 Filling and emptying the system . . . 6
2.1.4 Operation of the flow system . . . 9
2.2 Measurement hardware . . . 10
2.2.1 Laser Doppler Signal processing . . . 10
2.2.2 Traversing system . . . 11
2.2.3 Miscellaneous measurements . . . 11
2.2.4 Connections . . . 13

3 Software **16**
3.1 Menu system . . . 16
3.1.1 Starting the menu system . . . 16
3.1.2 Experimental setup . . . 17
3.1.3 File and data menu . . . 19
3.1.4 Trackers . . . 20
3.1.5 Pressure transducers . . . 22
3.1.6 ADC and amplifier menu . . . 25
3.1.7 Positioning of the laser . . . 25
3.1.8 Preparation menu . . . 28
3.1.9 Measurements . . . 28
3.1.10 Stepper motors . . . 30
3.2 Libraries . . . 31
3.2.1 LDADEC . . . 31
3.2.2 ADCLIB . . . 33
3.2.3 CFGLIB . . . 33
3.2.4 DPLIB . . . 34
3.2.5 FLOWLIB . . . 35
3.2.6 IOLIB . . . 36
3.2.7 LDALIB . . . 37
3.2.8 MATHLIB . . . 38
3.2.9 PARSLIB . . . 39
3.2.10 STEPLIB . . . 41

A File formats **42**
A.1 File format for processed data 42
A.2 File format of binary file for raw data 42

List of Figures

2.1 Schematic overview of pipe system 7
2.2 Two configurations of horizontal pipe section: straight and with contraction 7
2.3 Dimensions of the pipe contraction (sizes are in mm) 8
2.4 Configuration of the laser beams 11
2.5 Top view of the path of the laser beams 12
2.6 Traversing mechanism 13
2.7 Connections for the signal processing electronics. 14
2.8 Connections for the traversing system. 15
2.9 Connections for sensors measuring flow rate, water temperature and pressure difference 15

3.1 Configuration for calibration of pressure transducers. 24

1 Introduction

This report describes the hardware and software of the pipe flow system of the Fluid Dynamics Laboratory at Eindhoven University of Technology. It is not meant to be exhaustive, it should rather serve as a collection of practical knowledge about the setup. For scientific details about the flow system and the Laser Doppler Anemometry system the reader is referred to the thesis of Steenbergen (1995).

The pipe system was originally designed for the study of the decay of swirling pipe flow. Later on it has been adapted to study the combined effect of swirl and axial strain.

The water flow in the pipe system is driven by the pressure difference between the constant head tank on the roof (about 10 meters above the actual measurement pipe) and the basin in the basement (about 4 meters below). The horizontal pipe in which measurements can be made has a total length of $21m$. The flows that can be studied vary in two aspects. In the first place, the flows can either be swirling flows or not. A guide vane swirl generator is mounted at the inlet of the horizontal pipe and can be configured in different ways as to generate flows with different shapes of the tangential velocity profile (concentrated vortex, wall jet) and different swirl numbers (up to about 0.8). Secondly, either a straight pipe with a diameter of $70mm$ can be used or a contraction ($70mm \rightarrow 40mm$) can be mounted after some length of $70mm$-pipe. Then about $4m$ of $40mm$-pipe can be installed downstream of the contraction. The transition back to a pipe of $70mm$ is made through a vessel. The pipes of both diameters are divided in pieces of different lengths. Due to this modularity the configuration of the system, in terms of the axial position of the contraction, as well as the measurement sections (see below), is flexible.

Velocity measurements (both mean flow and turbulent quantities) can be made using a 2-dimensional Laser Doppler Anemometry (LDA) system. The optics can be traversed to make measurements at any radial position in the pipe. Through the combination of measurements with the optics mounted under three different angles (but perpendicular to the pipe axis) the full velocity vector as well as the full Reynolds stress tensor can be obtained. Optical access to the flow is possible through specially designed measurement sections, in which the pipe wall has been replaced by a thin transparent sheet. These measurement sections are available both with a pipe diameter of $70mm$ and $40mm$.

In the following chapters the actual pipe flow system is described as well as the equipment used to operate it (chapter 2). This report describes the situation as it was in the 'Warmte en Stroming' building, from which the Fluid Dynamics laboratory moved in the spring of 1999. Secondly the software used to control the measurements is described in some detail (chapter 3). The latter chapter also gives information on the procedures used in the operation of the flow system (alignment, calibration, etc.)

For more information the reader is referred to:

For further questions:

- Wiendelt Steenbergen (W.Steenbergen@tn.utwente.nl; knows nearly all);
- Gerald Oerlemans (pipe system, mechanics);
- Jan Voskamp (PEP-system, electronics, history);

- Pim Sluijter (optics);
- Gerard Trines (PEP-system, electronics);
- Arnold Moene (afmoene@hp1.met.wau.nl; present author).

2 Hardware

In this chapter the hardware of the pipe flow system is described: the actual flow system (section 2.1), the electronics and its connections (section 2.2).

2.1 Flow system

For more details on the pipe system the reader is referred to Steenbergen (1995).

2.1.1 Generation of flow

The pipe system is shown schematically in figure 2.1. The pumps are used to keep the water level in the upper tank constant (within about $5cm$). The flow rate is primarily controlled by operation of valve 2 (the use of a submerged valve proved to be advantageous with respect to cavitation). Valve 1 is used only occasionally for fine tuning of the flow rate. In the given configuration the maximum attainable flow rate is of the order of 60 m^3/hr (which corresponds to a bulk velocity of $4.3ms^{-1}$ for a pipe with inner diameter $70mm$).

Four pumps are available to pump the water from the basin in the basement to the reservoir on the roof. The pumps are switched on automatically when the water level in the reservoir has dropped below a prescribed level (the upper and lower level of the water surface differ by about $5cm$).

2.1.2 Pipe sections

The pipe sections used for the measurements consists of brass pipe with a wall thickness of $5mm$. The inner diameter is either $40 \pm 0.1mm$ or $70 \pm 0.1mm$, where the uncertainties are better than those required by the DIN standard (the given uncertainties have been measured at the ends of the pipe sections). The pipe sections are connected by specially designed couplings. These couplings ensure that pipe walls of two coupled pipes are aligned within $0.05mm$. Two flow configurations can be studied: a straight pipe with one diameter or a pipe with varying pipe diameter (see figure 2.2). The measurement section with internal diameter $70mm$ has three chambers with optical access (each 2 times $70mm$ apart), whereas the $40mm$ measurement section has two chambers (4 times $40mm$ apart).

The availability of sections of pipe for both diameters is summarised in table 2.1. Figure 2.3 shows the dimensions of the pipe contractions used in the studies on strained turbulence.

2.1.3 Filling and emptying the system

When the pipe system needs to be modified (e.g. measurement section to be installed at different axial position), the system needs to be emptied, at least between valve V4 and the swirl generator (see figure 2.1). In practice, the system is emptied between valve V3 and V4. The water is allowed to leave the system (into the sewer) through a tap in the bottom of the pipe, upstream of the swirl generator (while valves V3 and V4 are closed). A tap just below valve V3 needs to be opened in order to allow air to enter the system. When the water level has dropped such that air is entering the horizontal pipe section, air should be allowed into the chambers of the measurement section that surround the part of

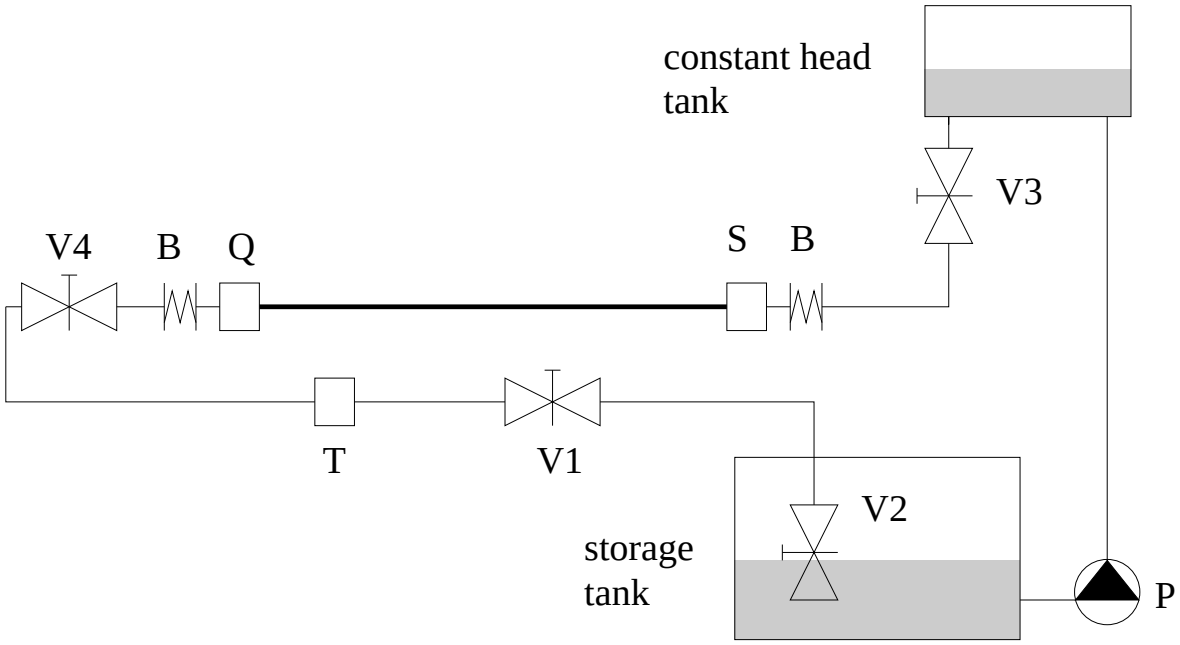

Figure 2.1: Schematic overview of pipe system (after Steenbergen (1995)). S: swirl generator; Q: flow meter; T: temperature sensor; V1, V2, V3 and V4: valves; B: rubber bellows; P: pumps (four times).

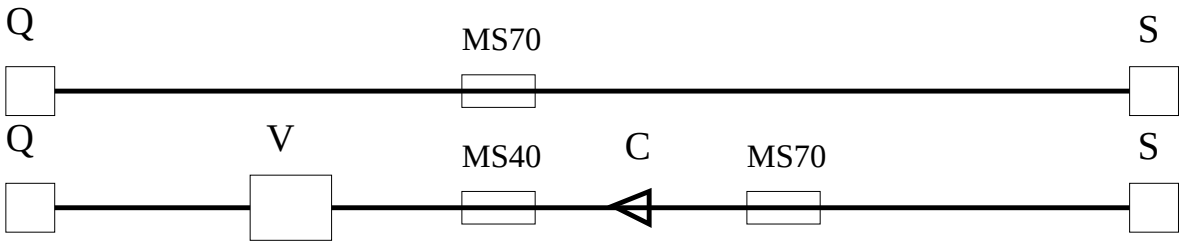

Figure 2.2: Two configurations of horizontal pipe section: straight pipe with a diameter of $70mm$ (top) or varying pipe diameter: $70mm \rightarrow 40mm \rightarrow 70mm$ (bottom). S: swirl generator; Q: flow meter; C: contraction; MS40 and MS70: measurement sections with internal diameter $40mm$ and $70mm$ respectively; V: vessel.

length (m)	$D = 70mm$	$D = 40mm$
0.31		1
0.369	1	
0.374	1	
0.475	1	
0.81		1
0.95	4	3
2.85	1	
3.80	1	
4.70	1	

Table 2.1: Overview of the sections of pipe available for both diameters. The fundamental unit length is $0.95m$ and most sections have a length which is a multiple of this.

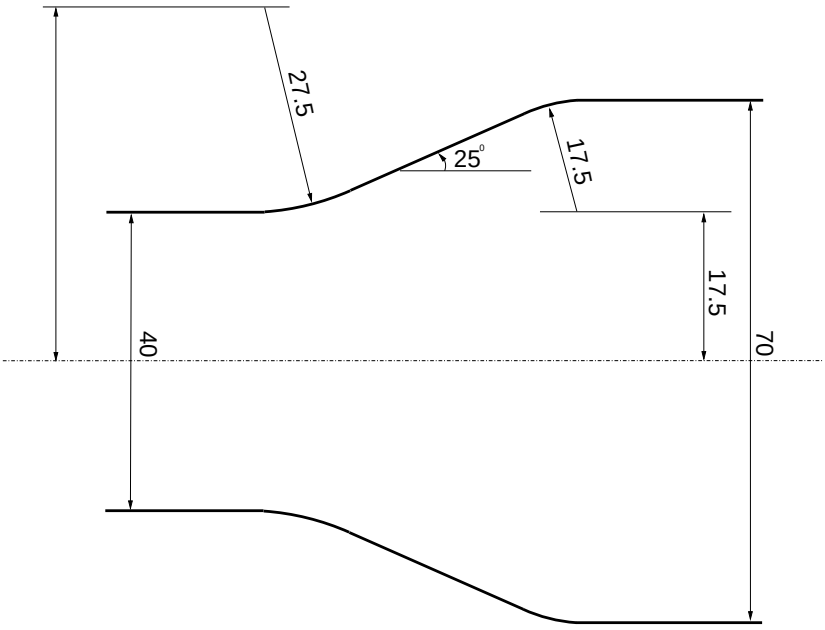

Figure 2.3: Dimensions of the pipe contraction (sizes are in mm)

the pipe wall replaced by film. Water can exit from the chambers through a small pipe system attached to the bottom of the measurement section. This small pipe system is also used to keep the water in the chambers at a slightly lower pressure than the water in the pipe, by connecting the small system to a the main pipe system at a position slightly downstream of the measurement section (where the static pressure is lower due to the pressure drop needed to keep the water flowing against friction). To allow air in, the bolts at the top of the chambers of the measurement section should be opened.

The system is refilled with water from a desalination unit. This water is fed into the system though a tap that is located near the thermometer T. In order to actually fill the system, valve V4 needs to be open, as well as the tap below valve V3 (to let the air escape from the system). While the horizontal pipe section is filling, water should be allowed into the chambers of the measurement section. This is done through a small pipe system attached to the measurement section. The water needed for this is drawn from the same supply that is used to fill the entire system. The water level in the chambers should more or less follow the level in the pipe. Once the pipe has filled completely, one should swiftly fill the entire chamber and make sure that no air is entrapped (air can be removed with a syringe to which a small tube is attached). To let the air escape more easily, the measurement section should be installed slightly inclined during the filling operation: the air bubbles will rise to one opening at the top of the chambers. Once the water has reached the level of valve V3 (and will flow out of the tap just below V3, which was left open to let air escape), the tap near the desalination unit needs to be closed. Then close the tap below valve V3 and open valve V3 slowly. Some stretch of vertical tube needs to be filled with water, so in order to prevent large pressure peaks in the pipe system, valve V3 should be opened very carefully. After filling has been completed, the system should run for some time in order to smooth out differences in temperature and composition between the 'old' and the 'new' water.

Since during the cycle of emptying and filling of the system water which contains seeding material is lost to the sewer, from time to time new seeding needs to be added to the water.

2.1.4 Operation of the flow system

In order to make measurements the water should flow through the pipe system. Usually, one needs to study a given flow at a given Reynolds number. This Reynolds number is attained in the following way:

- Switch on the pumps
- Open valve V3
- Open valve V4
- Open valve V1
- Open valve V2
- Let the water flow for at least 15 minutes in order to smooth out any difference in temperature etc.
- Use the software to show the Reynolds number on screen
- Adjust valve V2 such that the required Reynolds number is roughly attained (one will need some help from someone who will give the values of the Reynolds number, or one needs a display of the Reynolds number or flow rate near valve V2)
- Fine tuning of the Reynolds number can be done with valve V1

To stop the flow, one should take the following steps:

- Close valve V2
- Close valve V1
- Close valve V4
- Close valve V3
- Switch off the pumps

2.2 Measurement hardware

In this section we describe the hardware that is used to perform the measurements. On one hand this comprises the LDA optics and electronics and the data logging equipment needed to store the data (section 2.2.1). The LDA optics is positioned relative to the pipe using a traversing mechanism (see section 2.2.2). Then the miscellaneous measurements needed to characterise the flow are described (section 2.2.3) and finally the connections between the different electronic components is given in section 2.2.4.

The control of most components of the experimental setup, as well as the data-logging is performed using the PHYDAS system (Voskamp *et al.*, 1989).

2.2.1 Laser Doppler Signal processing

The velocity measurements are made using an LDA system of Dantec. The system is based on the reference beam principle (thus using the forward scattered light). The laser that is used is a 15 mW He-Ne laser manufactured by Uniphase. The optics mounted before the laser provide two reference beams and a scattering beam (see figure 2.4 and 2.5). The scattering beam is shifted in frequency ($40MHz$) by a Bragg-cell (the Bragg cell is driven by a signal from one of the frequency shifters Dantec 55N12). Two different front lenses can be mounted on the sending optics: one with a focal length (f) of 160mm and one with $f = 80mm$. The beams pass through the measurement section and the light of the reference beams and the light scattered in the direction of the reference beams is collected by lenses with a focal length of 60mm, passes through a pinhole and finally hits the photo diode (the pinhole is part of the housing of the diode Dantec 55L11). In the shifter the signal from the diode is shifted back by a frequency which is the sum of the $40MHz$ pre-shift of the Bragg-cell and a (positive or negative) frequency needed for an optimal use of the range of the tracker. From the shifter the signal is passed to the Dantec tracker 55N21) where the high frequency information is converted to a voltage (between 0 and 10 V).

The analog signal from the tracker is digitised using the parallel sampler (PARSAM) developed by 'Technical Laboratory Automation Group' at TUE (see Smeets and Nijmweegen (1993)). Since the PARSAM has an input voltage range of $-10V$ to $10V$, a programmable amplifier (built by Jan Niessen) is used to re-scale the signal from the tracker (see section 3.1.6). The output signal of the lock indicator of the tracker is fed directly to the PARSAM. The PARSAM uses a 12bit analog-to-digital converter (ADC). The choice to use a PARSAM in this experiment (rather than a conventional ADC) is based on two considerations:

- the PARSAM allows high sampling rates (up to 25 kHz per channel);
- two memory buffers (512 kB each) allow continuous operation, so that large amounts can be gathered continuously;

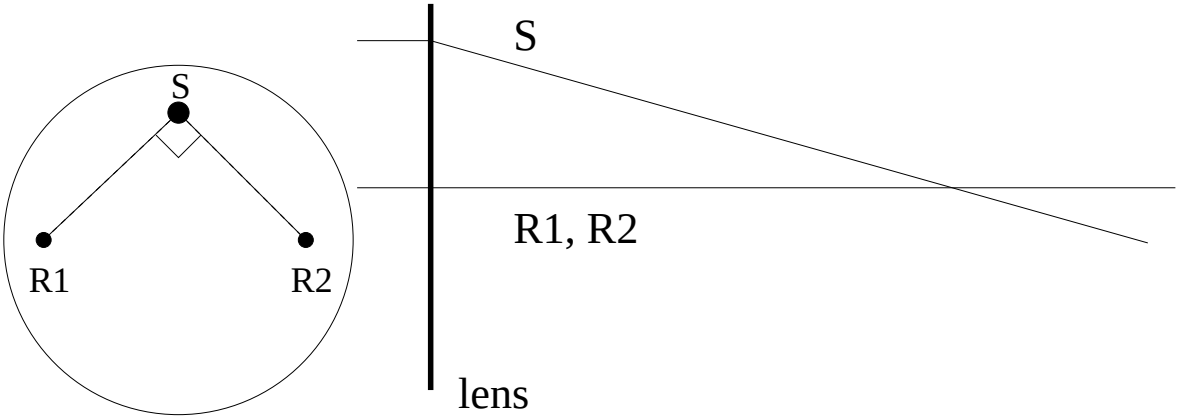

Figure 2.4: Confi guration of the laser beams, front view (left), side view (right): S denotes the scattering beam and R1 and R2 are the two reference beams.

These two characteristics are mainly important when information on the temporal structure of the signal is required (such as spectra).

2.2.2 Traversing system

The traversing system to move around the LDA optics is shown in figure 2.6. The plate to which the LDA optics are mounted can be turned around an axis that is parallel to the pipe over an angle of -50^o to 50^o. This rotation mechanism in turn is mounted on a plate that can be moved horizontally (y) and the latter plate is mounted to a plate that can be moved vertically (z). This vertical movement is relative to a rigid frame ('3' in figure 2.6) that can be moved horizontally (in a direction parallel to the pipe) within the outer rigid frame. The latter is standing on the floor of the laboratory. For the y and z movement use is made of stepper motors that allow movements with a step of $0.02mm$ to be made. The number of steps is checked by position encoders connected to the shafts of the stepper motors. End-switches are used to restrict the movement of the plate with optics to a bounded region. For the vertical movement counter weights are used in order to drastically reduce the amount of power needed to traverse the optics vertically.

2.2.3 Miscellaneous measurements

Besides the velocity measurement using the LDA, some extra quantities are measured: water temperature and flow rate.

The water temperature is monitored using a thermal diode of which the signal is converted to a voltage signal using a Laumann temperature display. The flow rate is measured by an Altometer magnetic flow meter (SC 80 AS), which has a current output proportional to the flow rate. This current is converted to a voltage by passing the current through a resistor of $470\Omega \pm 1\%$. The temperature and flow rate signal are digitised using a 12 bit ADC.

Incidentally measurements of pressure drop have been done. To this purpose pressure taps (with a diameter of $0.5mm$) are present in the pipe wall. These taps can be connected to one of the two pressure transducers, Validyne DP15-30 or DP45-16, which have a range of $|\Delta p| \leq 880$ and $35mmH_2O$, respectively. The signal of these pressure transducers is passed through an amplifier to a conventional ADC.

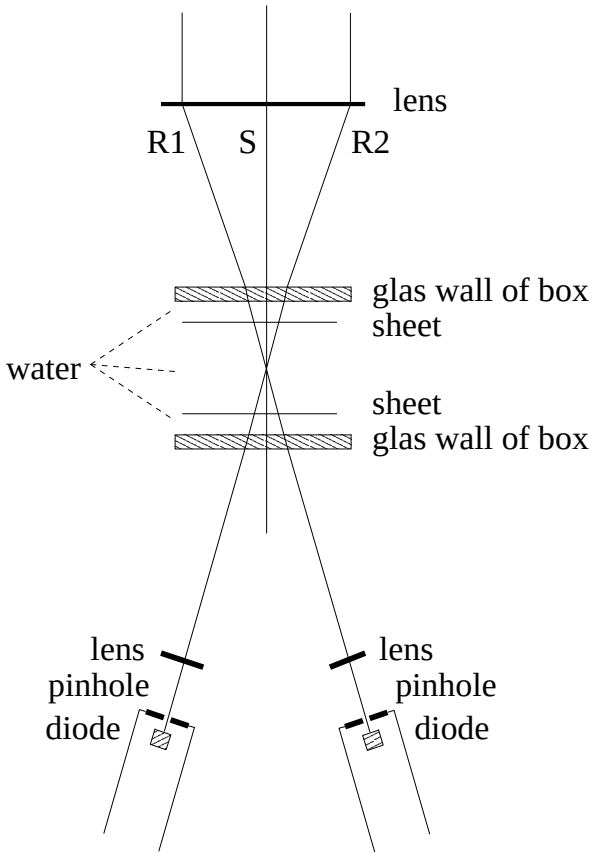

Figure 2.5: Top view of the path of the laser beams: from front lens of the sending optics, through the measurement section (glass walls and sheet) towards the diode optics

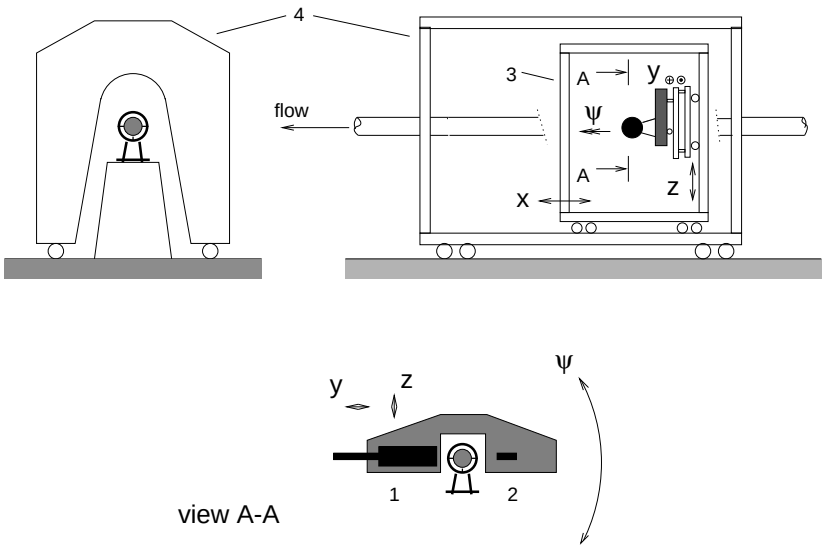

Figure 2.6: Traversing mechanism (figure from Steenbergen (1995)): view in the direction of the flow (top left), side view (top right) and the rotatable plate to which the optics are mounted (bottom); 1: transmitting optics; 2: photo-detectors; 3: rigid frame; 4: frame standing on floor; x-direction is flow direction, y is horizontal perpendicular to flow direction, z is vertical.

2.2.4 Connections

In figure 2.7 the connections between the different components of the LDA signal processing system are shown. For the calibration of the frequency trackers the shifter and tracker are disconnected and the output of the function generator is connected to the 'photo detector input' connectors of the trackers.

The connections between the components of the traversing system are shown in figure 2.8.

Figure 2.9 shows the connections between the different components that are used for the measurement of flow rate, water temperature and pressure drop. For the pressure drop measurements only one pressure transducer is used at a time. However, in principle it is possible to connect both transducers to the ADC (each at a different input channel) since both transducers have an individual amplifier.

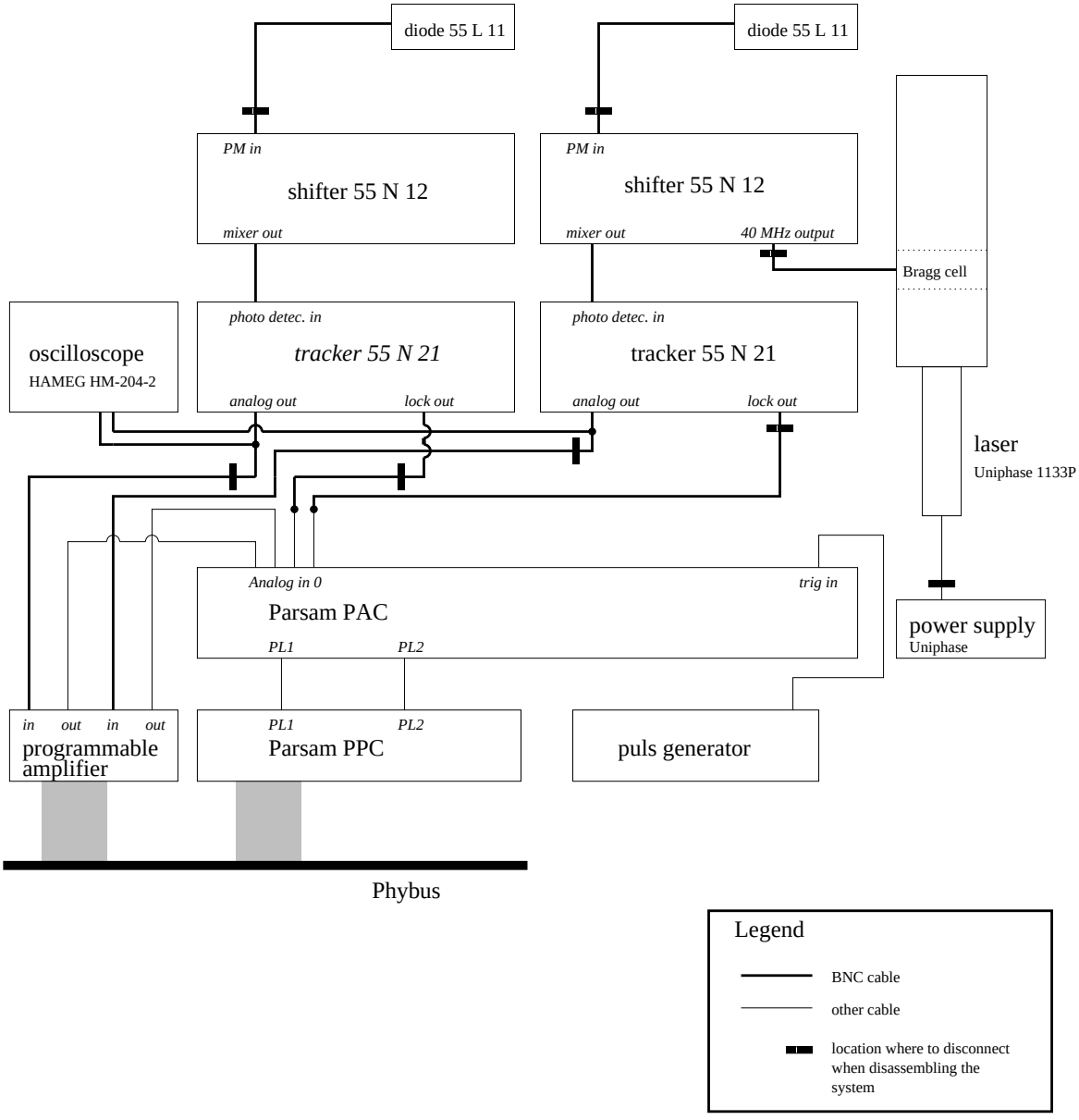

Figure 2.7: Connections for the signal processing electronics.

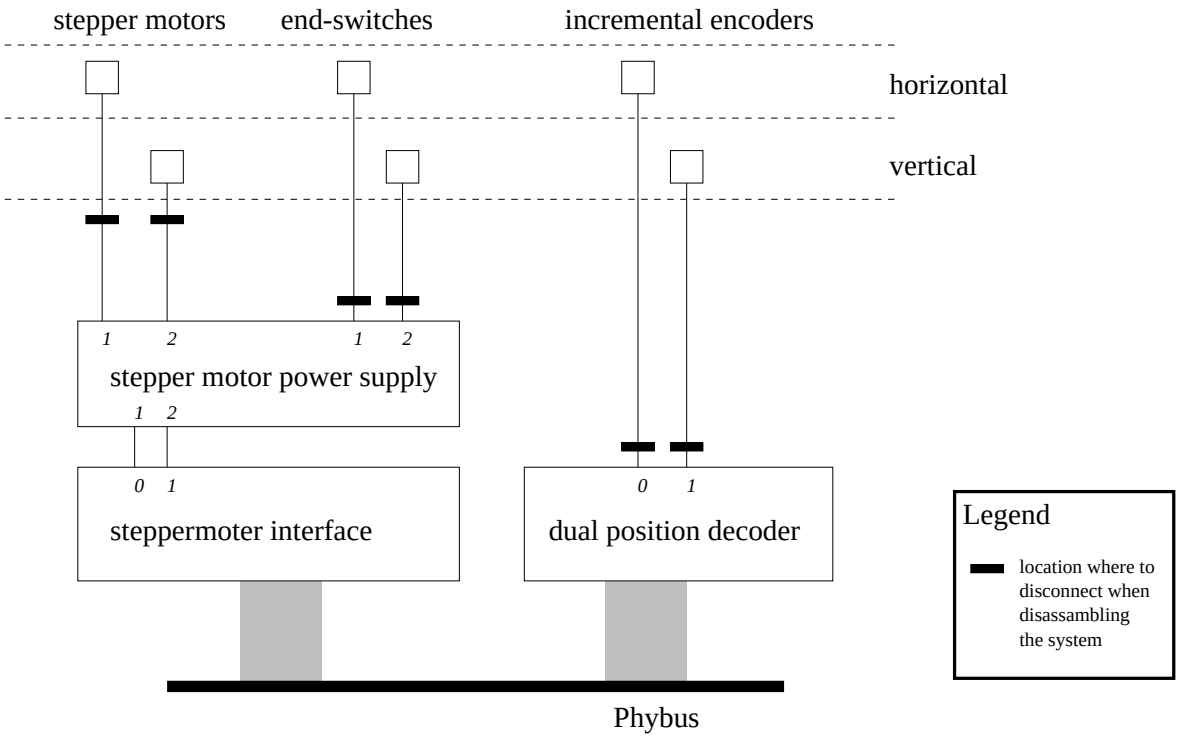

Figure 2.8: Connections for the traversing system.

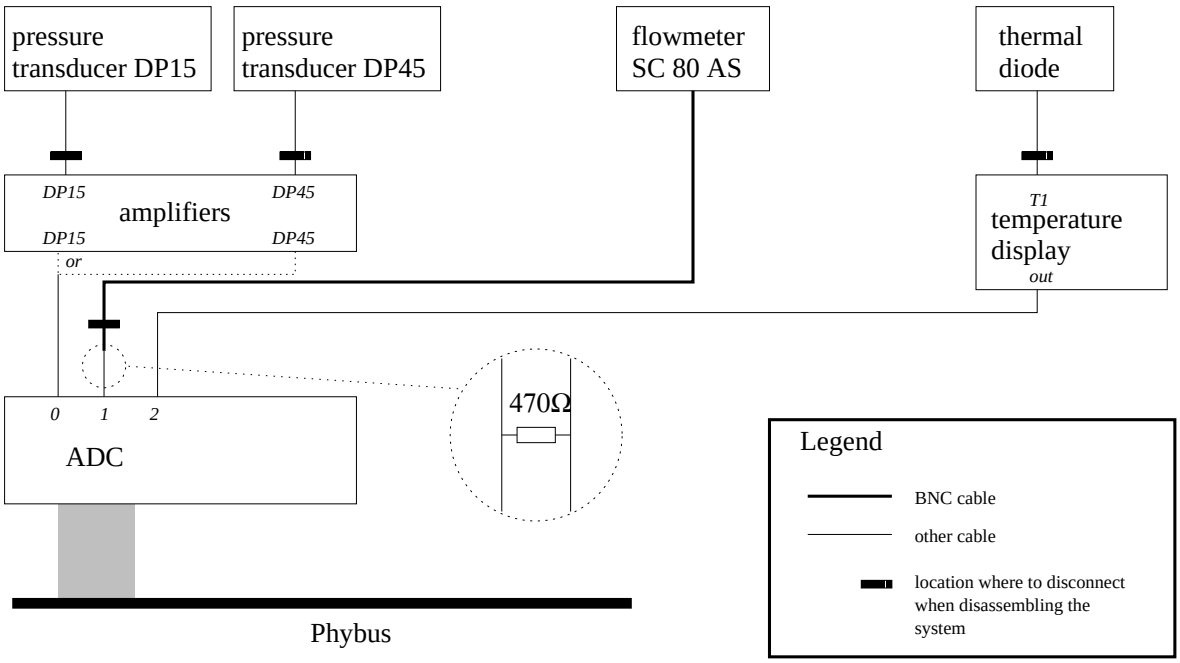

Figure 2.9: Connections for sensors measuring flow rate, water temperature and pressure difference. The resistor of 470Ω is used to convert the current output of the flow meter to a voltage output.

3 Software

3.1 Menu system

The programs to control the LDA setup have been united under one menu system. In order to use the menu system one needs to make use of a 'new' PEP version (newer or equal to PEP030 2.30.0). Furthermore it is assumed that a VT240 terminal emulator is used.

3.1.1 Starting the menu system

Wait to get the `MPS030>` prompt. Then load the PEP system (`QL MPS:PEP030TL.Q` and after completion: `RUN`). Then you are asked for the user ID (`User id?`) and answer with `GS` unless you have been given another ID. The next step is to load the monitor program: `MON HWD;ZELPAGE.HWC`. Finally you can automatically load all required libraries using the command `load_libs`, which asks you for a filename from which the library names should be read (give a carriage return to load the default file `GSD:lolibs.GSX` or type an alternative name. When you use the default, the following libraries will be loaded (in this order):

- `PEP:ST240.PTC`
- `GSD:PLOTVT.GSC`
- `GSD:ST7.GSC`
- `GSD:MATHLIB.GSC`
- `GSD:IOLIB.GSC`
- `GSD:LDADEC.GSC`
- `GSD:PARSLIB.GSC`
- `GSD:ADCLIB.GSC`
- `GSD:LDALIB.GSC`
- `GSD:FLOWLIB.GSC`
- `GSD:DPLIB.GSC`
- `GSD:STEPLIB.GSC`
- `GSD:CFGLIB.GSC`

After the libraries have been loaded you type `RUN MENU` to start the system. When the menu system was not terminated properly the previous time it was run, a temporary file may still be present (default name `gsd:tempfile.gst`). The program will ask whether you want to read settings and data from this file.

Next, the program will ask whether you want to switch on the stepper motors. If you answer `Y`, the program will store the readings of the position encoders. Then you will be asked to switch on the stepper motors and to subsequently hit a key. The program will now determine the displacement of the encoders due to the switching on of the stepper motors and correct for the displacement. This procedure is used to ensure that once the LDA optics have been aligned properly, this alignment can be used even after the PEP system has been switched off. For this reason it is also advisable that whenever the computer is switched off, the optics are brought in such a position that measurement volume is at the centre of the pipe (this corresponds to `y_pos = 0` and `z_pos = 0`).

After the menu system has started you will be presented with the main menu:

```
################################################################################
## Menu level 1                    Main menu                                  ##
################################################################################


OPTIONS : S    = Setup of experiment
          F    = Files
          T    = Trackers (calibration, ranges etc.)
          D    = Pressure transducer settings
          A    = ADC
          L    = Lining of LDA
          P    = Preparation of measurements
          M    = Measurements
          C    = Control of steppen motors
          O    = Open a monitor session
          Q    = Stop
OPTION : S
```

In the following sections we will discuss the contents of each of the sub menus. When you just have started the menu system, the usual first step is to go to the Experiment setup menu in order to define the experiment or to retrieve the setup from a file.

3.1.2 Experimental setup

The layout of this menu is shown below:

```
################################################################################
## Menu level 2               Experiment setup                              ##
################################################################################

OPTIONS : O   = Measure mode               (now: lda     )
          C   = Code for experiment        (now: mn076axx)
          SN  = New name for setup file    (now: gsd:setup.gsd)
          SA  = Save setup to gsd:setup.gsd
          SR  = Retrieve setup from gsd:setup.gsd
          N   = Number of samples          (now: 50000)
          F   = Sampling frequency         (now: 100 Hz)
          L   = Choose focal length of lens (now:   160. mm)
          A   = Choose angle of LDA        (now:   0.00 deg.)
          T   = Choose angle for traverse  (now:   0.00 deg.)
          P   = Change table with positions measuring volume
          M   = Measure pressure gradient  (now: OFF)
          Q   = Quit to main menu
OPTIE : O
```

When the current experiment does not differ too much from the previous, one usually starts with retrieval of the setup from a file of which the name can be set by the command `SN` (default name `GSD:SETUP.GSD`).

For the measurement mode (option `O`) one can choose between LDA measurements or pressure measurements:

```
##############################################################################
## Menu level 3                Measurement mode                            ##
##############################################################################


OPTIONS : 1    = lda
          2    = pressure
          Q    = Exit
OPTION: 1
```

The code of the experiment (option C) is used to generate the file name for the data file (and is also stored in the binary file used to store raw data). Sampling characteristics can be set with option F (sampling frequency) and N (number of samples). Please note that the sampling frequency is only used by the program to do some checks. The program can not set it, since the sampling frequency is selected by choosing the correct output of the pulse generator (the PARSAM is triggered externally).

One can choose between two lenses with different focal lengths ($80mm$ and $160mm$). This information is used by the program to convert the Doppler frequency to velocities (the beam geometry is different for the different lenses).

Furthermore, two angles can be set. The angle of the LDA (option A) is the angle of the LDA optics relative to the horizontal, where a positive angle corresponds to an anti-clockwise rotation, when looking in the flow direction. The traverse angle (option T) is the angle between the line through all positions to be measured and the horizontal (same sign convention as for the other angle). Usually, one uses zero for this angle (a horizontal traverse). One could use 90^o to make a direct measurement of the radial component. Or one could use measurements over a different traverse to study asymmetries in the flow.

The radial positions at which measurements are made are defined in the so-called position table. A separate menu is available to manipulate this table:

```
##############################################################################
## Menu level 3                Position table                              ##
##############################################################################


OPTIONS : TN  = Change name of position table file (now: gsd:posfil.gsd  )
          TR  = Retrieve position table from gsd:posfil.gsd
          TS  = Save position table to gsd:posfil.gsd
          E   = Edit position table
          C   = Create new position table
          S   = Show current position table
          P   = Set position pointer to specific value (now: 3)
          Q   = Stop
OPTION : E
```

Position tables can be stored in a file. The name of this file can be set with the option TN (default name gsd:posfil.gsd), the file can be saved with TS and retrieved with TR.

The position table consists of an array with two columns. One column contains radial position (expressed as r/R) and the other column contains a flag that signifies whether the measurement of velocity data has been completed successfully for that positions. Throughout the program the 'position pointer' is maintained: a number that points to the next position for which measurements will be done (in this menu it can be reset to a specific value (option P).

With the option S one can show the current position table on the screen.

Using the options E or C one can edit the position table. In the case of creating a position table the user is first asked whether the existing table may be destroyed. Issuing the options C or E will eventually lead one to the following menu:

```
##############################################################################
## Menu level 4               Edit position table                          ##
##############################################################################


OPTIONS : A1  = Add a point
```

```
          AM  = Add more than one point
          D1  = Delete one point
          DM  = Delete more than one point
          DS  = Delete successful points
          R   = Reset info on success of processing
          S   = Show table
          Q   = Stop
OPTION :A1
```

To add one point to the list, use option `A1`. The user can enter the r/R-value for the point, followed by a carriage return. If one needs to add more points, choose `AM`. The sequence of points entered must be ended by typing a `Q` rather than a number. To delete one or more points from the table, one first needs to know the positions of those points (this can be found out by showing the table on the screen: option `S`). With option `DS` all points for which the measurements have been completed successfully can be removed from the table.

When in the experimental setup menu the measurement mode (option `O` is set set to pressure, the menu looks like:

```
################################################################################
## Menu level 2                Experiment setup                              ##
################################################################################

OPTIONS : O   = Measure mode               (now: pressure)
          C   = Code for experiment        (now: mn076axx)
          SN  = New name for setup file    (now: gsd:setup.gsd)
          SA  = Save setup to gsd:setup.gsd
          SR  = Retrieve setup from gsd:setup.gsd
                Number of samples fixed to         400
          F   = Sampling frequency         (now: 100 Hz)
          Q   = Quit to main menu
OPTIE : Q
```

3.1.3 File and data menu

The file and data menu offers the possibility to define which type of LDA data should be collected and where and how it should be stored. It is accessed from the main menu through the option `F` and gives the following menu:

```
################################################################################
## Menu level 2              File and data menu                              ##
################################################################################


OPTIONS : DV  = change standard VAX device      (now: gsd)
          DL  = change standard LDN device      (now: gud)
          NVP = name processed data file on VAX (now: gsd:mn076axx.gsd)
          NLP = name processed data file on LDN (now: gud:mn076axx.gsd)
          SLR = Save raw data to LDN            (OFF)
          SLP = Save processed data to LDN      (ON)
          SVP = Save processed data to VAX      (ON)
          Q   = Quit to main menu
OPTION : SLR
          Raw data to LDN [y/n] ? y
```

The arrangement of a VAX device and a LDN (Local Data Net) device is based on the historical situation when most data and programs where stored on a VAX at the 'Technical Laboratory Automation Group' and a fast link using LDN was used within the Fluid Dynamics Laboratory. Nowadays still two separate systems exist, but the VAX has been replaced by a PC at the Fluid Dynamics Laboratory and the disk space it uses resides at the a DEC Alpha (`tnl.phys.tue.nl`, directory `/tnlu3/phylan/fti/`). No speed difference exists anymore between 'VAX' and LDN. The data written to LDN reside on `tnl.phys.tue.nl` as well, in the directory `/tnlu2/data`. The 'device' that can be set with options `DV` and `DL` are the names of the subdirectories in the directories mentioned before. The names of the output files are formed automatically from the device name and the experiment code (set in the setup menu, see 3.1.2). With options `SLR` and `SLP` one can choose the data to be saved at LDN: either raw data (binary files with time series of both trackers and optionally

Range	Lower frequency (kHz)	Upper frequency (kHz)
1	1	10
2	3	33
3	10	100
4	33	333
5	100	1000
6	300	3300
7	1000	10000

Table 3.1: Number of the tracker ranges.

output data of the lock channel of the trackers) or processed data (ASCII file of means, variances etc.). With SVP one can choose to store processed data on the VAX device.

When raw data are stored some extra options are given:

```
##############################################################################
## Menu level 2              File and data menu                             ##
##############################################################################


OPTIONS : DV  = change standard VAX device      (now: gsd)
          DL  = change standard LDN device      (now: gud)
          NVP = name processed data file on VAX (now: gsd:mn076axx.gsd)
          NLR = name raw data file on LDN       (now: gud:mn076axx.gsr)
          NLP = name processed data file on LDN (now: gud:mn076axx.gsd)
          SLR = Save raw data to LDN            (ON)
          SLP = Save processed data to LDN      (ON)
          SVP = Save processed data to VAX      (ON)
          L   = Store tracker lock data         (ON)
          B   = Block type                      (contin. 1 block)
          R   = Nominal LDN rate (byte/sec)     (    45000.)
          Q   = Quit to main menu
OPTION : B
          1. contin. 1 block
          2. contin. more blocks
          3. non-contin. blocks

          Block type : 2
```

One can request to store the output of the lock-indicators of the trackers (option L). With option B the way in which raw data are written to file can be chosen. Option 1 (continuous, one block) implies that all data of a times series are written to file in one operation. This requires that all data can be stored in the memory ($4Mb$ in the present setup) of the MPS system at once. If more data will be gathered than can be stored at once, one can chose to write the data in blocks. After one part of the PARSAM memory ($512kB$) has been filled and new samples are stored in the other part of the memory, the data from the first part of memory are written. However, this only works when the data can be written to disk faster than that they are gathered in the other PARSAM memory. Therefore one needs to set a nominal speed of the LDN-connection (option R). If the sampling rate is too high to allow the data to be written to file in time, one has to choose option 3: non-continuous blocks. This will result in chunks of time series that are not contiguous in time. Whether the choice for a given block type is realistic will be tested before the measurements start.

3.1.4 Trackers

Option T in the main menu brings the user to the tracker menu. Here one can choose the settings for the trackers. The range can be choosen by number (see table 3.1).

Some guidance in the choice of the tracker range can be given with te following rules:

- The Doppler frequency is for each reference beam is proportional to $\frac{1}{2}\sqrt{2}$ of the flow velocity along the pipe axis (due to the configuration of reference and scattering beam, see figure 2.4). Consequently, if the mean flow velocity along the pipe axis is denoted by U, the Doppler frequency to be expected is $f = \frac{1}{2}\sqrt{2}U\frac{n}{\lambda}\sin\theta$, where n is the refractive index of water (1.33), λ is the wavelength of the light used (632.8 nm) and θ is the half angle between the scattering beam and the reference beam (for the $f = 160mm$ lens this is about 8^o). Thus the mean Doppler frequency is about $2.1 \cdot 10^5 Hz/ms^{-1}$.

- The above rule can be summarized by writing the relationship between frequency and velocity in vector notation:

$$\begin{pmatrix} f_1 \\ f_2 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \end{pmatrix} \tag{3.1}$$

  where f_1 and f_2 are the Doppler frequencies for the two trackers, U_1 is the velocity component parallel to the pipe axis and perpendicular to the optical axis of the laser, U_2 is the velocity component perpendicular to the optical axis of the laser and perpendicular to the plane formed by the two reference beams, a_{11}, a_{12} and a_{21} are matrix components with values of about $2.1 \cdot 10^5 Hz/ms^{-1}$ and a_{22} is about $-2.1 \cdot 10^5 Hz/ms^{-1}$.

- The tracker range should be chosen large enough to allow for large turbulent velocity fluctuations.

- The frequency shift should be chosen such that the low mean velocities occuring in the wall layer are within the range of the tracker, whereas the zero-velocity of the wall is outside the tracker range. In some cases these two requirements can not be made simultaneously (one could think of very low wall layer mean velocities with large turbulent fluctuations, including backflow).

The tracker menu is shown below.

```
################################################################################
##  Menu level 2                  Trackers                                    ##
################################################################################

OPTIONS : R1   = Set range tracker 1 (now 5)
                 (calib. date: 26-3-1998  8:45:43 )
          R2   = Set range tracker 2 (now 5)
                 (calib. date: 26-3-1998  8:45:43 )
          S1   = Set shift frequency for tracker 1 (now 20000)
          S2   = Set shift frequency for tracker 2 (now 20000)
          C    = Calibrate trackers
          D    = Delete all calibrations
          Q    = Stop
OPTIE : R1
        Give range index of tracker 1 : 5
```

The pre-shift frequency can be choosen for each channel separatly as well. Possible values comprise $f_s = A \times B \times CkHz$, where $A = 1$ or -1, B is any integer value between 1 and 9 and C is either 10, 100 or 1000.

```
################################################################################
##  Menu level 2                  Trackers                                    ##
################################################################################

OPTIONS : R1   = Set range tracker 1 (now 5)
                 (calib. date: 26-3-1998  8:45:43 )
          R2   = Set range tracker 2 (now 5)
                 (calib. date: 26-3-1998  8:45:43 )
          S1   = Set shift frequency for tracker 1 (now 20000)
          S2   = Set shift frequency for tracker 2 (now 20000)
          C    = Calibrate trackers
          D    = Delete all calibrations
          Q    = Stop
OPTIE : S2
        Give shift for tracker 2 (in Hz) : 20000
```

Calibration of the trackers can be done with option C, which will yield the following menu.

```
################################################################################
## Menu level 3                  Calibration                                  ##
################################################################################

OPTIONS : R    = Set other than default range (default: 5, current : 5)
          C    = Calibrate trackers
          Q    = Stop
OPTIE : C
```

Option R gives the opportunity to calibrate aan other range than the default one. Option C starts the actual calibration:

The program asks to offer three different frequencies to the trackers. This is done by connecting the output of the function generator to the 'photo detector input' of the tracker. An amplitude of the signal of 0.01 V is sufficient. A useful setting of the trigger frequency for the PARSAM is 1000 Hz. The frequencies to be applied depend on the seletced range of the tracker and the settings of the gain and amplitude of the amplifiers.

```
################################################################################
## Menu level 4             Calibration of tracker                            ##
################################################################################

Offer to tracker a frequency of    100000. Hz
Hit any key
V1  = 1.0600375
V2  = 1.0386424
dV1 =-6.0037494e-02
dV2 =-3.8642406e-02

Offer to tracker a frequency of    250000. Hz
Hit any key
V1  = 2.6264081
V2  = 2.5793993
dV1 =-.12640810
dV2 =-7.9399347e-02

Offer to tracker a frequency of    450000. Hz
Hit any key
V1  = 4.6954336
V2  = 4.6109991
dV1 =-.19543362
dV2 =-.11099911

Hit any key
```

Finally, option D in the tracker menu can be used to delete all calibrations:

```
################################################################################
## Menu level 2                  Trackers                                     ##
################################################################################

OPTIONS : R1   = Set range tracker 1 (now 5)
                 (calib. date: 20-7-1998 14: 5:18 )
          R2   = Set range tracker 2 (now 5)
                 (calib. date: 20-7-1998 14: 5:18 )
          S1   = Set shift frequency for tracker 1 (now 20000)
          S2   = Set shift frequency for tracker 2 (now 20000)
          C    = Calibrate trackers
          D    = Delete all calibrations
          Q    = Stop
OPTIE : D
```

3.1.5 Pressure transducers

Option D in the main menu makes one enter the pressure transducer sub menu:

```
################################################################################
## Menu level 2             Pressure transducers                              ##
################################################################################


OPTIONS : T    = Set transducer to be used (now none)
          C    = Calibrate transducer
          D    = Delete all calibrations
          P1   = Position of first pressure point (now:  0.000 m)
          P2   = Position of second pressure point (now:  0.000 m)
          R    = Set index in result array (now:            0)
```

```
          Q    = Stop
OPTIE : T
          1 = DP15
          2 = DP45
          Give number of transducer : 1
```

One can choose (option T) between two transducers, DP15 and DP45, which have a range of $|\Delta p| \leq$ 880 and $35mmH_2O$, respectively. In order to calibrate the transducer select option C:

```
################################################################################
## Menu level 3                  Calibration                                 ##
################################################################################


OPTIONS : O    = Set other than default transducer
            (default: DP15, current : DP15)
          C    = Calibrate pressure transducer
          Q    = Stop
OPTIE : C
```

Here you can choose to calibrate another transducer than the one currently selected for measurements (option O) or to proceed with the calibration. An example (with fake numbers) of a calibration is given below.

```
################################################################################
## Menu level 4       Calibration of pressure transducer                     ##
################################################################################

Open shunt valve; have water level difference of approx. 0 cm
Hit any key
V =  4.3223446e-04
Close shunt valve; do not move bottle
Give exact height (in cm!)  : 1
Hit any key
V  = 4.3711846e-04
Set water level to approx. -44.0 cm
Give exact height (in cm!)  : 2
Hit any key
V  = 4.7619047e-04
Set water level to approx. -79.2 cm
Give exact height (in cm!)  : 3
Hit any key
V  = 5.2991451e-04
 a =  .26985410     b = -942.91589     c =  752190.19
```

The pressure difference used for the calibration is applied using two small bottles, connected to the transducer through plastic tubes (see figure 3.1). One of the bottles is at a fixed height (in a standard on the table). The other is mounted on a vertical scale with a nonius along which it can be slided. First one has to determine the voltage at zero pressure difference (with open shunt valve). Then one keeps both bottles at the same height and closes the shunt valve to obtain the first actual reading. Subsequently, the bottle at the vertical scale is moved to two lower positions. Approximate heights are indicated by the program, depending on the transducer to be calibrated. Exact heights to a precision of $0.01mm$ must be entered. The final result are three coefficients in a second order polynomial $\Delta p = a + bV + cV^2$.

After the calibration the pressure transducer menu is returned to. Here one can set the axial position (in m) of the two pressure taps in the pipe. Furthermore, the array index in the result array where the next measurement will be stored can be set:

```
################################################################################
## Menu level 2              Pressure transducers                            ##
################################################################################


OPTIONS : T    = Set transducer to be used (now DP15)
            (calib. date: 3-6-1998 )
          C    = Calibrate transducer
          D    = Delete all calibrations
          P1   = Position of first pressure point (now:  0.000 m)
          P2   = Position of second pressure point (now:  0.000 m)
          R    = Set index in result array (now:           0)
          Q    = Stop
OPTIE : R
        Give new index in results array  : 1
```

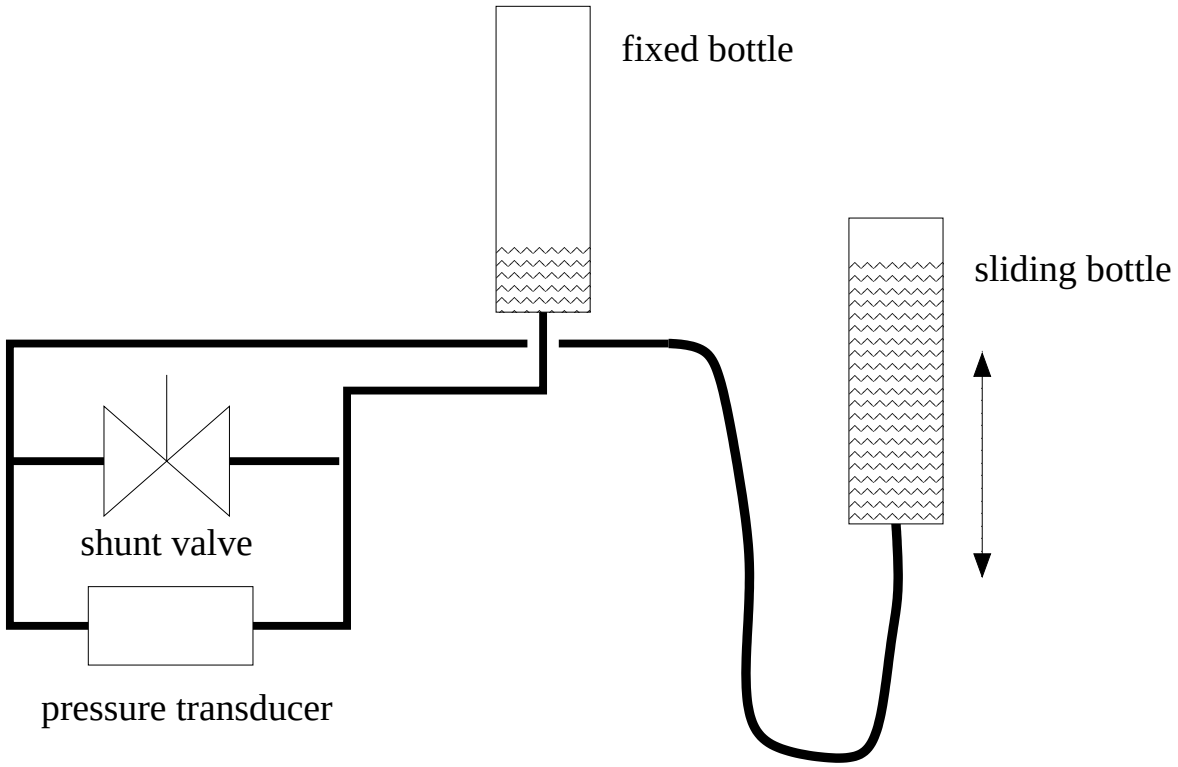

Figure 3.1: Confi guration for calibration of pressure transducers.

If the measurement mode is set to pressure measurements (in the setup menu, see 3.1.2), the pressure transducer menu looks like:

```
################################################################################
## Menu level 2              Pressure transducers                            ##
################################################################################

OPTIONS : T    = Set transducer to be used (now DP15)
            (calib. date: 3-6-1998 )
          C    = Calibrate transducer
          D    = Delete all calibrations
          P1   = Position of first pressure point (now:  0.000 m)
          P2   = Position of second pressure point (now:  0.000 m)
          R    = Set index in results table (now:     44)
          M    = Do measurement
          S    = Show results
          Q    = Stop
OPTIE : Q
```

Extra options are `M` to actually do a pressure measurement with these settings and `S` to show the results. Thus the procedure to do pressure measurements is:

- Calibrate the transducer to be used.
- Connect the tubes of the transducer to the pressure taps to be used and flush any air from the tubes.
- Set the axial positions of the taps used with options `P1` and `P2`.
- Do the measurement (option `M`).
- Repeat the last three steps for all combinations of pressure taps required.

3.1.6 ADC and amplifier menu

Selecting option A in the main menu gives the ADC and amplifier menu. Here one can choose which ADC (Analog-to-Digital Converter) one wishes to use (option C), either PARSAM, or the so-called 'old ADC':

```
################################################################################
##  Menu level 2            ADC and amplifier setup                         ##
################################################################################

OPTIONS : C   = Choose type of ADC (now: PARSAM)
          G   = Choose gain for amplifier (now: 4)
          O   = Choose offset for amplifier (now:   2.50 V)
          P   = Choose gain for amplifier of PARSAM (now: 1)
          Q   = Quit to main menu
OPTIE : C
         PARSAM (P) or old ADC (O) : P
```

Option G gives the possibility to set the gain of the programmable amplifier. This is used to re-scale the output of the tracker ($0 - 10V$) to the input range of the PARSAM ($-10 - 10V$):

```
################################################################################
## Menu level 3               Amplifier gain menu                           ##
################################################################################

OPTIONS:  0 : multiplier =  1.0000000
          1 : multiplier =  2.0000000
          2 : multiplier =  4.0000000
          3 : multiplier =  6.0000000
          4 : multiplier =  8.0000000
          5 : multiplier =  10.000000
          6 : multiplier =  15.000000
          7 : multiplier =  20.000000
Give index of required gain: 1
```

The gains given in the menu are nominal values, whereas in the actual data processing exact, measured gain values are used. Besides, the program ensures that the calibration of the trackers is performed with the same settings of gain and offset for the amplifier as the actual measurements. In the ADC and amplifier menu the option O lets one set the offset of the amplifier (the offset is applied *before* the gain). For rescaling of the tracker output to make it usable for the PARSAM, useful combinations of gain and offset are 2 and $5V$ or 4 and $2.5V$, respectively. Finally, the PARSAM has a build in amplifier as well, which can be set through option P.

3.1.7 Positioning of the laser

Option L in the main menu will produce the menu 'Positioning of the laser'. Besides the two options given here (horizontal alignment –option H– and determination of the centre of the pipe –option C–) the total alignment entails more steps:

- The glass walls of the measurement section need to be exactly vertical. This is checked by placing a brass plate on top of the measurement section and a precision water level on top of the brass plate [1].
- The traversing system needs to be (horizontally) parallel to the pipe axis. This is checked as follows:
  - Rotate the laser and its optics around the optical axis over 90 degrees, so that the reference beams describe a vertical plane.
  - Let the reference beams illuminate the glass wall of the measurement section. If the beams are perpendicular to the glass wall, the beams and their reflections will lie in one plane.

[1] For measurements where the optics are under an angle of ± 45 degrees, a block with an angle of exactly 45 degrees can be placed on top of the brass plate.

– Assuming that the laser is mounted in the traversing system such that the optical axis is perpendicular to the axial dimension of the traversing system, the latter will now be parallel to the pipe axis.

- The traversing system needs to be exactly horizontal (both in the direction of the pipe and in the direction perpendicular to it). This is checked with a water level. The four feet on which the traversing system rests can be adjusted independently.
- The laser beams need to be adjusted such that the velocity $v_1 + v_2$ (resulting from the reference beams 1 and 2, respectively) is exactly parallel to the pipe axis and $v_1 - v_2$ is exactly perpendicular to it (option `H`)
- The coordinate system of the pipe needs to be linked to that of the traversing system at one point: the pipe centre (option `C`).

```
##############################################################################
## Menu level 2              Positioning of laser                           ##
##############################################################################


OPTIONS : H   = Horizontal lining of laser
          C   = Find centre of pipe
          Q   = Stop
OPTION : H
```

Choosing for the horizontal alignment gives the following menu:

```
##############################################################################
## Menu level 3           Horizontal lining of laser                      ##
##############################################################################

OPTIONS : L   = Perform horizontal lining
          C   = Calibrate tracker range 2 (calibrated)
          Q   = Stop
OPTION : L
```

For the horizontal alignment of the laser, a brass plate is mounted on top of the measurement section (as for the horizontal placement of the measurement section itself). On top of the brass plate a perspex platelet is placed. This will serve as a scattering object with a known direction of its velocity (viz. horizontal). The laser optics have to be moved such that the measurement volume coincides with the (vertical) face of the perspex platelet at the side of the laser. The LDA velocity measurements will be performed with tracker range 2 and the menu offers the option to calibrate the tracker at this range. Furthermore one has to ensure that the shift frequency set at the shifter is identical to the one set in the tracker menu (see 3.1.4, a useful value is $10000Hz$). A useful sampling frequency for these measurements is $1000Hz$ (1000 samples are taken). The actual alignment procedure implies six measurements of the velocity of the platelet (which is moved by hand, in the forward –flow– direction). Only measurements with sufficient quality (viz. a minimum number of samples for which the tracker locked) are used. After the six measurements, a graph is shown on the screen, showing the relationship between the 'horizontal' and 'vertical' (in laser optics coordinates) velocity on the horizontal and vertical axes. The slope of the line gives the slope of the laser optics relative to the horizontal. Compute the arctangent of the slope to obtain the angle over which the laser optics should be turned around the optical axis [2]. The ultimate goal is to reduce the slope such that the misalignment is less than 0.1 degree. This alignment does not need to be repeated when the laser optics are turned for measurements at ± 45 degrees.

Option `C` in the alignment menu gives the sub-menu for determining the centre of the pipe.

[2] To get rid of the graph on the screen press `<RETURN>`, followed by `Ctrl-G`.

```
################################################################################
## Menu level 3            Centre and radius of pipe                          ##
################################################################################


OPTIONS : S    = Start procedure for pipe centre
          R    = Set radius of pipe manually (now 20.00 mm )
          Q    = Stop
OPTION : S
```

With option R one can set the radius of the (which is a by-product of this alignment) manually to the nominal value (either 20*mm* or 35*mm*). It is good practice to do this, after the determination of the centre of the pipe. For this alignment, the laser optics is moved such that the measurement volume is in the neighbourhood of the pipe centre. The laser is turned on, but the scattering beam is not needed. Using the telescope mounted on top of the plate where the diodes are mounted, the horizontal positions (in *mm*) of the intersections of the two reference beams with the pipe wall are measured. This is done first with the measurement volume near the pipe centre (and for the intersections at the diode-side of the pipe and at the laser-side of the pipe):

```
################################################################################
## Menu level 4               Find centre of pipe                             ##
################################################################################
Situate measuring volume near pipe centre


OPTIONS : Unnnn = move nnnn steps up
          Dnnnn = move nnnn steps down
          Fnnnn = move nnnn steps forward
          Bnnnn = move nnnn steps backward (to wall)
          C     = continue
OPTION : C
Beam/wall intersection at diode side
Position left intersection ? : 2.45
Position right intersection ? : 3.54
distance =  4.8745079
wall_pos[1,0] = -3.4467914
wall_pos[1,1] =  3.4468038
Hit any key

Situate measuring volume near pipe centre


OPTIONS : Unnnn = move nnnn steps up
          Dnnnn = move nnnn steps down
          Fnnnn = move nnnn steps forward
          Bnnnn = move nnnn steps backward (to wall)
          C     = continue
OPTION : C
Beam/wall intersection at laser side
Position left intersection ? : 2.23
Position right intersection ? : 3.43
distance = -4.8745079
wall_pos[2,0] =  3.4467914
wall_pos[2,1] = -3.4468038
Hit any key
```

Subsequently, the measurement volume is moved to an off-axis position (vertically):

```
################################################################################
## Menu level                 Find centre of pipe                             ##
################################################################################
Situate measuring volume off centreline


OPTIONS : Unnnn = move nnnn steps up
          Dnnnn = move nnnn steps down
          Fnnnn = move nnnn steps forward
          Bnnnn = move nnnn steps backward (to wall)
          C     = continue
OPTION : C
Beam/wall intersection at diode side
Position left intersection ? : 3.48
Position right intersection ? : 5.15
distance =  9.7490158
wall_pos[3,0] = -6.8935828
wall_pos[3,1] =  6.8936076
Hit any key
```

After this final measurement the program will report the distance from the pipe centre of the current origin of the coordinate system of the traversing system. And more importantly it gives an estimate of the pipe radius. This can be used to check the accuracy of the alignment. The computed pipe radius should differ from the nominal value by less than $0.1mm$.

3.1.8 Preparation menu

Some preparatory tasks, before the measurements are started, can be performed in the preparation menu (option P in the main menu):

```
################################################################################
## Menu level 2          Preparation for measurements                        ##
################################################################################

OPTIONS : C   = Check LDA signal at positions in table
          I   = Initialize values of flow, temp. ...
          D   = Display Reynolds number (OFF)
          Q   = Stop
OPTION : Q
```

Option C enables the user to check the quality of the LDA signal at the radial positions stored in the position table (edited in the setup menu, see 3.1.2). Option I initializes the arrays with measurements of flow rate and temperature, used for the computation of the Reynolds number. With option D one can toggle the display of the Reynolds number, flow rate and temperature on the screen. This display is updated every two seconds and can be used to adjust the flow rate to obtain the correct Reynolds number.

3.1.9 Measurements

To start the measurements, select option M in the main menu. Before the measurements are started, a number of settings are checked:

```
################################################################################
## Menu level 2           Checking before measurements                       ##
################################################################################
    OK  Processed data to VAX, file name has been set ?
    OK  Processed data to LDN, file name has been set ?
    OK  Raw data tot LDN, file name has been set ?
    OK  Output to any type of file ?
    OK  Has tracker range been set ?
    OK  Has pre-shift been set ?
    OK  Has range been calibrated ?
    OK  Calibration for correct ADC ?
    OK  Has amplifier gain been set ?
NOT OK  Calibration done with gain set to current value ?
    OK  Has amplifier offset been set ?
    OK  Calibration done with with offset set to current value ?
    OK  Has PARSAM gain been set ?
    OK  Calibration done with PARSAM gain set to this value ?
    OK  Has angle of traverse been set ?
    OK  Has angle of LDA been set ?
    OK  Has pipe centre been determined ?
Hit any key
```

If anything is not correct, the user has to go back and correct the error. Once the first series of checks is passed, some other checks are performed:

```
################################################################################
## Menu level 2                    Checking                                  ##
################################################################################

Checking sampling frequency ...

- Trigger frequency appears to be 100 Hz
- Data written to file at once
  Checking speed of connection via LDN ...
- Transfer to LDN  44943.820     byte/s
  This is SLOWER than the nominal value  45000.000
Hit any key
```

The actual sampling frequency is checked against the one selected in the setup menu (see 3.1.2). Then the block type of for raw data storage is checked (if raw data storage has been selected in the file menu, see 3.1.3). Finally, the actual speed of the LDN connection is checked against the nominal value given in the file menu (if raw data storage has been selected).

After all tests have been passed, the following menu appears:

```
################################################################################
## Menu level 2                Measurements menu                           ##
################################################################################


OPTIONS : S   = Start automatic measurements
          M   = Start manual measurements
          X   = Max. multiple of required samples to get (now:  2.0)
          L   = Lock limit (now:  2200)
          P   = Set position pointer to specific value (now 3)
          R   = Set current index in results table (now 44)
          F   = Flag point  0.0000000    as done
          D   = Toggle display of Re, temp. and flow (now: OFF)
          Q   = Stop
OPTION : M
```

Option `S` starts the automatic LDA measurements. In the case of processed data storage this implies the following steps:

- The program moves the optics to the radial position given in the position table (starting at the array index given after option `P`).
- The program determines the quality of the signal:
  - A limited number of samples is taken.
  - If for less than 75% (`sfrac` in the program) of these samples the tracker was locked (locked: the lock signal was above the lock limit, given after option `L` in raw ADC output), the number of samples to be taken is increased in order to retain the originally requested number of samples. If this final number of samples if more than the maximum multiple given following option `X`, the position is skipped for further measurements.
  - If for more than 75% of the samples the tracker was locked the original number of samples will be gathered.
- If the position is not skipped, the measurements will be made. The quality of the data will be assessed again (the program requires the tracker to have locked for more than 75% of the samples). If the data quality is sufficient the data will be processed and stored to file. Internally a flag will be set that this point was completed successfully (which can be done manually with option `F`. Note that the processed data are stored in ASCII format, but in a binary file: EPEP does not permit the writing of lines longer than 128 characters to an ASCII file.
- After completing measurements at a given point, the program proceeds to the next point in the position table.

For the raw data the procedure is simpler: no check on the quality of the data is made: data are taken and written to file. The format of this binary file is given in appendix A.2.

When manual measurements are chosen (option `M`), data are taken for one position at a time, and the user can either cycle through the positions stored in the position table or give other values for r/R.

3.1.10 Stepper motors

The stepper motors can be controlled from the stepper motor menu (option C in the main menu.

```
################################################################################
## Menu level 2           Control of stepper motors                           ##
################################################################################


OPTIONS : O    = Switch on stepper motors
          S    = Make steps in y- or z-direction
          R    = Make steps to r/R position
          C    = Move to pipe centre
          Y    = Set y_pos to a value (now: 0)
          Z    = Set z_pos to a value (now: 0)
          Q    = Stop
OPTION : S
```

Option O is equivalent to what happens when one starts the menu system and the program asks whether one wants to switch on the stepper motors. The program will store the readings of the position encoders. Then one will be asked to switch on the stepper motors and to subsequently hit a key. The program will now determine the displacement of the encoders due to the switching on of the stepper motors and correct for the displacement. This procedure is used to ensure that once the LDA optics have been aligned properly, this alignment can be used even after the PEP system has been switched off. For this reason it is also advisable that whenever the computer is switched off, the optics are brought in such a position that measurement volume is at the centre of the pipe (this corresponds to `y_pos = 0` and `z_pos = 0`).

Option S gives the following menu:

```
################################################################################
## Menu level 3        Move steps in y- and z-direction                       ##
################################################################################

OPTIONS : Unnnn = move nnnn steps up
          Dnnnn = move nnnn steps down
          Fnnnn = move nnnn steps forward
          Bnnnn = move nnnn steps backward (to wall)
          Q     = stop
OPTION : Q
```

The steps mentioned in the menu are steps of $0.02mm$, so there are 500 steps in a cm.

To make a move to a specific radial position in the direction of the traverse (given in the setup menu, see 3.1.2), choose option R in the stepper motor menu:

```
################################################################################
## Menu level 2           Control of stepper motors                           ##
################################################################################

OPTIONS : O    = Switch on stepper motors
          S    = Make steps in y- or z-direction
          R    = Make steps to r/R position
          C    = Move to pipe centre
          Y    = Set y_pos to a value (now: 0)
          Z    = Set z_pos to a value (now: 0)
          Q    = Stop
OPTION : R
         Give r/R value : 0
```

Option C is equivalent to choosing $r/R = 0$ for option R. Setting the y-position and z-position (options Y and Z, respectively), does not move the laser optics, but just shifts the internal coordinate system. Use with care: one might loose the correct alignment !!

3.2 Libraries

A number of special purpose libraries (in EPEP) have been developed for use with the menu system described in the previous section. The subroutines and functions provided by these libraries are described in the following sections.

3.2.1 LDADEC

The library LDADEC is loaded before all other libraries that where designed specifically for the pipe flow system. LDADEC provides all global variables for those libraries and for the menu system that is loaded on top of the libraries. The documentation of each library mentions whether it needs LDADEC to be loaded or not.

```
%.....................................................
% Library    :  ldadec
% Purpose    :  To declare (and partly initialize) all global
%               variables used by the LDA system.
% Required libraries:
%               GSD:iolib.GSC
% Required compiled code:
%               none
% Note       :  Library test only for PEP030 2.30.0
% Revisions:    13-06-95: storage of raw data on VAX has been
%               disabled in other libraries, but relevant
%               variables still present in LDADEC
%               13-06-95: defines for types of raw data storage
%               replace store_at_once and contig_block
%.....................................................
DECLARE
  % length of array with processed data
  len_data = 20

  % dummy value
  no_value = 999

  % address of ADC for LDA measurements
  adc_lda = 1040

  % address of ADC for flow and temperature measurement
  adc_flow = 1024

  % count value above which lock output should be
  % for tracker to be locked
  lock_limit: nat2

  % measurement mode
  measmode: integer
  modecode: ARRAY 3, 8 OF char
  % defines for type of storage of raw data (raw_store_type)
  old_adc_lock = 0
  con_1_lock = 1
  con_n_lock = 2
  nocon_n_lock = 3
  old_adc_nolock = 50
  con_1_nolock = 51
  con_n_nolock = 52
  nocon_n_nolock = 53
  store_lock = 0
  nostore_lock = 50
  old_adc = 0
  con_1 = 1
  con_n = 2
  nocon_n = 3

  % name of file for backup of setup
  temp_fname = 'gsd:tempfile.gst'

  % maximum number of samples that can stored on VAX
  % in one write statement (put_array is limiting
  % factor) (see boolean store_at_once)
  vax_maxsample = 2**16

  % Name of setup file
  setup_fname,

  % Name of file with radial positions
  postable_name,

  % Data file names
  ldnraw_name,
  ldnproc_name,
  vaxraw_name,
  vaxproc_name: ARRAY 16 OF char

  % default device names on VAX and LDN
  vaxdev,
  ldndev: ARRAY 3 OF char

  % Switches
  % OK to measure (i.e. all checks OK)
  measure_ok,

  % PARSAM current ADC ?
  parsam_in_use,

  % Name of setup file set ?
  set_setup_name,

  % Name of position table file set ?
  set_postable_name,

  % Name of data files set ?
  set_vaxproc_name,
  set_ldnproc_name,
  set_vaxraw_name,
  set_ldnraw_name,

  % Name of default devices set ?
  set_vaxdev_name,
  set_ldndev_name,

  % Focal length of lens set ?
  set_lens,

  % Angle of LDA set ?
  set_angle_lda,

  % Angle of traverse set ?
  set_angle_trav,

  % Tracker range set ?
  set_track_range,

  % Shift frequency set ?
  set_track_offset,

  % ADC gains or offset set ?
  set_pars_gain,
  set_amp_gain,
  set_amp_offset,

  % Store raw and/or processed data to VAX and/or LDN
  set_rawvax
  set_procvax,
  set_rawldn
  set_procldn,

  % Sampling freqency set ?
  set_freq,

  % Number of samples set ?
  set_sample,

  % Centre of pipe has been determined ?
  set_pipe_centre,

  % Experiment code been set ?
  set_expcode: boolean

  % Calibration information
  % Date of calibration for each range
  calib_date: ARRAY 1..7 OF integer

  % Time of calibration for each range
  calib_time: ARRAY 1..7 OF integer

  % PARSAM used as ADC during calibration ?
  calib_pars: ARRAY 1..7 OF boolean

  % Amplifier gain used during calibration
  calib_gain: ARRAY 1..7 OF real
```

```
% Amplifier offset used during calibration
calib_offset: ARRAY 1..7 OF real

% PARSAM gain used during calibration
calib_pgain: ARRAY 1..7 OF real

% Has range been calibrated ?
calib: ARRAY 1..7 OF boolean % tracker calibrated for range ?

% Current sampling frequency
freq: integer

% Positions of pressure points (up- and downstream)
dpx1, dpx2: real

% Have dpx1 and dpx2 been set ?
set_dpx1, set_dpx2: boolean

% Do pressure measurements ?
press_meas: boolean

% Which transducer in use ?
press_trans: nat1

% Names of pressure transducers
press_name: ARRAY 3, 4 OF char

% Full scale range of transducers (in m H2O)
press_full_scale: ARRAY 1..2 OF real

% Has pressure transducer been choosen
set_press_trans: boolean

% Has transducer been calibrated
calib_press: ARRAY 1..2 OF boolean

% Date of calibration
calib_press_date: ARRAY 1..2 OF integer

% Calibration coefficients (V = a + b*dp + c*dp*dp)
press_coeff: ARRAY 1..2, 1..3 OF real

% Current number of samples
sample,

% Current maximum number of samples in integer array
% (from PARSAM): length of array is 0.5 * max_intsamp
max_intsamp: integer

% Current maximum number of samples if nat2 array
max_nat2samp: integer
max_nat2locksamp: integer

% Nominal speed of LDN connection (in bytes/sec)
nom_ldnspeed: real

% Way in which raw data are stored (and which)
raw_store_type: integer

% Number of signals
n_signal: integer

% Experiment code
expcode: ARRAY 0..7 OF char

% Current menu level
menu_level: nat1

% Current range number for both trackers (index 1 and 2)
rangeindex: ARRAY 0..2 OF nat1

% Current frequency shift on preshifter
f_s: ARRAY 1..2 OF integer

% Maximum frequency in tracker range (7 ranges)
def_range: ARRAY 1..7 OF integer % max. freq. in tracker range

% Nominal gains of amplifiers used with old ADC
mp_nom: ARRAY 8 OF real % nominal gain factors for amplifiers

% Table with real gains of amplifiers used with old ADC
mp_table: ARRAY 1..2, 8 OF real

% Conversion matrix : freq. -> speeds
scf: ARRAY 1..2, 1..2 OF real

% Calibration (2nd order) of trackers
coeff: ARRAY 42 OF real

% Transformation matrix from discrete stepper motor
% coordinates to continuous pipe coordinates (and v.v.)
trmatr, invtrmatr: ARRAY 1..2, 1..2 OF real

% Array with processed data of each radial point
res_table: ARRAY 1..128, len_data OF real

% Names of variables in res_table
res_name: ARRAY len_data, 8 OF char

% Number of points in position table (pos_table)
npoint,

% Number of current observation in result table (res_table)
res_pt,

% Index in pos_table of current point
pt: integer

% Current point in pressure results array
presspoint: integer

% Table of radial positions (r/R) where measurements
% will be done; table whether processing was succesful
pos_table: ARRAY 1..128 OF real
pos_success: ARRAY 1..128 OF boolean

% Current amplifier offsets and gains (per channel)
os: ARRAY 0..2 OF real
mp: ARRAY 0..2 OF real

% PARSAM gain
p_gain: ARRAY 0..2 OF real

% Current y- and z-position in stepper motor coordinates
y_pos, z_pos: int2

% Should (when processing data) 3rd moments be
% calculated
do_3mom: boolean

% Leave ldadec (and loose values of global variables) ?
bewaren: boolean

% Current value of focal length of lens
flens: real

% Angle of LDA (relative to horizontal)
lda_angle: real

% Angle of traverse (relative to horizontal)
trav_angle,

% Angle between reference beams
beam_angle,

% Current value of pipe radius
radius: real

% Answer to question
answer: boolean

BEGIN
  % (1) Define ranges
  FOR i IN 1..7 DO calib[i] := false OD
  def_range[1] := 10000
  def_range[2] := 33000
  def_range[3] := 100000
  def_range[4] := 333000
  def_range[5] := 1000000
  def_range[6] := 3300000
  def_range[7] := 10000000

  % (2) Set amplifier gains
  % (2.1) Real gains
  mp_table[1, 0] := 1.00
  mp_table[1, 1] := 1.98
  mp_table[1, 2] := 3.95
  mp_table[1, 3] := 5.94
  mp_table[1, 4] := 7.89
  mp_table[1, 5] := 9.85
  mp_table[1, 6] := 14.89
  mp_table[1, 7] := 19.78
  mp_table[2, 0] := 1.00
  mp_table[2, 1] := 1.99
  mp_table[2, 2] := 3.96
  mp_table[2, 3] := 5.95
  mp_table[2, 4] := 7.92
  mp_table[2, 5] := 9.89
  mp_table[2, 6] := 14.97
  mp_table[2, 7] := 19.88
```

```
  % (2.2) Nominal values
  mp_nom[0] := 1
  mp_nom[1] := 2
  mp_nom[2] := 4
  mp_nom[3] := 6
  mp_nom[4] := 8
  mp_nom[5] := 10
  mp_nom[6] := 15
  mp_nom[7] := 20

  % (3) Set defaults
  % (3.1) Parsam is ADC
  parsam_in_use := true
  mp[1] := mp_table[1, 0]
  mp[2] := mp_table[2, 0] set_amp_gain := false
  os[1] := 0
  os[2] := 0 set_amp_offset := false
  p_gain[1] := 1
  p_gain[2] := 1 set_pars_gain := false
  n_signal := 2
  lock_limit := 2200

  % (3.2) Set name of default setup file
  strcpy('gsd:setup.gsd', setup_fname)
  set_setup_name := true

  % (3.3) Initially not OK to measure (check first)
  measure_ok := false

  % (3.4) Do not compute 3rd moments
  do_3mom := false

  % (3.5) Initialize PHYBUS
  init_pb

  % (3.6) Set mode
  measmode := 1
  modecode[1] := 'lda     '
  modecode[2] := 'pressure'
  % (4) Give way to other libraries
  bewaren := true
  WHILE bewaren DO
    monitor
    cls
    inq_bool(20, 20, 'Do you want to leave LDADEC
                     and loose data',
      answer)
    IF answer THEN bewaren := false
    ELSE bewaren := true FI
    cls
  OD
END
```

3.2.2 ADCLIB

```
%.................................................
% Library  :  adclib
% Purpose  :  Contains routines for old ADC's
% Author   :  Arnold Moene (this library); others
%             (ADC procedures)
% Required libraries:
%             none
% Required compiled code:
%             GUD:adcopepn.GUO
% Note     :  Library tested only for PEP030 2.30.0
%.................................................
```

meet_adc_old3, meet_adc_old2, meet_adc_old1

```
%.................................................
% Procedures: meet_adc_old3, ...2 and ...1
% Purpose   : To get 3, 2 or 1 arrays of nat2 from an ADC
% Interface : a, b, c    IN/OUT  arrays to store data
%             address    IN      start PHYBUS address of ADC
%             num_chan   IN      number of channels to sample
% Note      : The number of samples to get is determined from
              the size of a, b, or c
%.................................................
PROCEDURE meet_adc_old3(a, b, c: ARRAY 0 OF nat2, address,
                        num_chan: nat2) AT adc_code[0]
PROCEDURE meet_adc_old2(a, b: ARRAY 0 OF nat2, address,
                        num_chan: nat2) AT adc_code[0]
PROCEDURE meet_adc_old1(a: ARRAY 0 OF nat2, address,
                        num_chan: nat2) AT adc_code[0]
```

adc_meas

```
%.................................................
% Procedures: adc_meas
% Purpose   : To get a samples from a given channel of an ADC
%             on a given PHYBUS address
% Interface : address    IN      start PHYBUS address of ADC
%             chan       IN      channel to sample
%             return value       nat2 samples from requested channel
%.................................................
PROCEDURE adc_meas(address: nat2, chan: nat2): nat2
```

loop_adc

```
%.................................................
% Procedure: loop_adc
% Purpose  : To continuously get samples from a number of channels
% Interface: addr   IN   addres of adc
%             nchan  IN   number of channels (always starting at
%                         the first
%.................................................
PROCEDURE loop_adc(addr: nat2, nchan: nat2)
```

offset, gain

```
%.................................................
% Procedures: offset and gain
% Purpose   : To set offset and gain of amplifiers
% Interface : swa    IN     number of channel (later converted to
%                           addres of amplifier
%             g0     IN     gain
%             to     IN     offset
%.................................................
PROCEDURE offset(swa, to: int2)
PROCEDURE gain(swa, g0: int2)
```

get_adc_char

```
%.................................................
% Procedure: get_adc_char
% Purpose  : To get ADC characteristics (volts per count and
%            counts at zero voltage)
% Interface: parsam_adc  IN   is PARSAM ADC of which you want
%                             characteristics ?
%            v_per_c     OUT  volts per count
%            zero_c      OUT  counts at zero voltage
%.................................................
PROCEDURE get_adc_char(parsam_adc: boolean,
                       v_per_c: REF real,
                       zero_c: REF nat2)
```

3.2.3 CFGLIB

```
%.................................................
% Library  :  cfglib
% Purpose  :  Routines used for the configuration of the
%             experiment
% Author   :  Most routines have been written by
%             Wiendelt Steenbergen. Modifications and
%             new routines are due to Arnold Moene
% Date     :  1993-1995
% Required libraries:
%             HWD:st5.HWC
%             GSD:mathlib.GSC
%             GSD:iolib.GSC
%             GSD:ldadec.GSC
% Note     :  Library tested only for PEP030 2.30.0
%.................................................
```

set_raw_type

```
%.................................................
% Procedure  :  set_raw_type
% Purpose    :  To set raw_store_type to a value,
%               according to values of number of parameters
% Interface  :  block_d  IN    one or more, (non)continous
%                              blocks ?
%               lock_d   IN    store lock data ?
%               return value   whether valid type has
%                              been set
%.................................................
PROCEDURE set_raw_type(block_d, lock_d: integer): boolean
```

get_raw_type

```
%...................................................
% Procedure  :  get_raw_type
% Purpose    :  To get from raw_store_type the values
%               for a number of parameters
% Interface  :  block_d  OUT   one or more, (non)continous
%                              blocks ?
%               lock_d   OUT   store lock data ?
%               return value   whether valid values have
%                              been gotten
%...................................................
PROCEDURE get_raw_type(block_d, lock_d: REF integer): boolean
```

eval_raw_type

```
%...................................................
% Procedure  :  eval_raw_type
% Purpose    :  To evaluate whether the requested raw
%               data storage type is possible (with
%               respect to MPS-memory and LDN-speed)
% Interface  :  return value     a boolean, indicating
%                                whether the storage type
%                                has been changed
%...................................................
PROCEDURE eval_raw_type: boolean
```

calc_maxsamp

```
%...................................................
% Procedure  :  calc_maxsamp
% Purpose    :  To compute the maximum number of samples
%               in the nat2 and/or integer arrays used for
%               sampling (in measmen)
% Note       :  Most variables used here are globals,
%               declared in ldadec
%...................................................
PROCEDURE calc_maxsamp
```

order_pos

```
%...................................................
% Procedure  :  order_pos
% Purpose    :  To sort a position table and remove double
%               or deleted items (having value no_value).
% Interface  :  table     IN/OUT    position table
%               succ      IN/OUT  success flags
%               n         IN/OUT    length of table
%...................................................
PROCEDURE order_pos(table: REF ARRAY 0 OF real,
                    succ: REF ARRAY 0 OF boolean,
                    n: REF nat2)
```

midden_straal

```
%...................................................
% Procedure  : midden_straal
% Purpose    : To determine coordinates (in LDA reference
%              frame) of centre of pipe.
% Interface  : pos1    IN    measured relative position
%                            of intersection of film and
%                            reference beams
%              pos2    IN    idem
%              pos3    IN    idem
%              midden  OUT   y-z position of centre of
%                            pipe
%...................................................
PROCEDURE midden_straal(pos1, pos2, pos3: ARRAY 2 OF integer,
                        midden: REF ARRAY 2 OF integer)
```

lens_choice

```
%...................................................
% Procedure  :  lens_choice
% Purpose    :  To update parameters that depend on the
%               focal length of the lens used.
% Interface  :  f    IN    focal length of lens
%...................................................
PROCEDURE lens_choice(f: real)
```

calc_transf_matrix

```
%...................................................
% Procedure  :  calc_transf_matrix
% Purpose    :  To determine the transformation matrix
%               from the discrete stepper motor frame
%               (y_pos, z_pos) to the continuous
%               reference frame of the pipe (y, z)
% Interface  :  lda_a    IN    angle of LDA, relative
%                              to horizontal.
%...................................................
PROCEDURE calc_transf_matrix(lda_a: real)
```

save_setup

```
%...................................................
% Procedure  :  save_setup
% Purpose    :  saves value of most global variables
%               to a file
% Interface  :  fname    IN    name of file to save to
%...................................................
PROCEDURE save_setup(fname: CONST string)
```

retrieve_setup

```
%...................................................
% Procedure  :  retrieve_setup
% Purpose    :  To read values of most global variables
%               from a file
% Interface  :  fname    IN    name of file to read from
%...................................................
PROCEDURE retrieve_setup(fname: CONST string)
```

3.2.4 DPLIB

```
%...................................................
% Library  :  dplib
% Purpose  :  Procedure related to measurement and use of
%             pressure gradients.
% Author   :  Wiendelt Steenbergen (with some modifications
%             by Arnold Moene)
% Date     :  1993-1995
% Required libraries:
%             GSD:ldalib.GSC
%             GSD:iolib.GSC
%             GSD:mathlib.GSC
%             HWD:st7.HWC
%             GSD:adclib.GSC
%...................................................
```

cal_corr

```
%...................................................
% Procedure : cal_corr
% Purpose   : To correct the calibration factors of the
%             pressure transducer for the fact that the
%             first reading was not at zero pressure
%             difference. The voltage measured with open
%             shunt valve is used for the correction.
% Interface : a, b, c   IN/OUT  calibration coefficients
%                               in dp=a+b*V+c*V*V
%             shunt     IN      voltage when shunt valve
%                               is open
%             no_shunt  IN      voltage when shunt valve
%                               is closed
%...................................................
PROCEDURE cal_corr(a, b, c: REF real, shunt, no_shunt: real)
```

meas_press

```
%...................................................
% Procedure  : meas_press
% Purpose    : To measure the output voltage of the
%              pressure transducer
% Interface  : press_sample    IN   number of samples to use
%              avg_dpv         OUT  output voltage
%...................................................
PROCEDURE meas_press(press_sample: integer, avg_dpv: REF real)
```

ustar

```
%...................................................
% Procedure: ustar
% Purpose  : To determine u* from a given pressure gradient
%            (in mH2O/m) and a radius (in m)
% Interface: pressg     in   pressure gradient (in mH2O/m)
%            rad        in   radius (in m)
```

```
%            return value    u* (in m/s)
%.................................................
PROCEDURE ustar(pressg, rad: real): real
```

yplus

```
%.................................................
% Procedure: yplus
% Purpose  : To compute y+ from a given radial position
%            (in r/R), pressure gradient, radius and
%            temperature
% Interface: pressg   In   pressure gradient (mH2O/m)
%            rr       IN   r/R (-)
%            rad      in   radius (m)
%            temp     in   temperature (deg C)
%            return value  y+ (-)
%.................................................
PROCEDURE yplus(pressg, rr, rad, temp: real): real
```

press_grad

```
%.................................................
% Procedure  : press_grad
% Purpose    : To determine the pressure gradient between
%              two points (of which the positions are given
%              by dpx1 and dpx2) in (m H2O)/m
% Method     : The distance is determined from the global
%              variables dpx1 and dpx2, the reading
%              of the pressure transducer is taken from
%              dp_vao_end (measured in process flowtemp)
%              and the pressure is determined from dp_vao_end
%              and the calibration factors for the transducer
%              used.
% Interface  : return value       pressure gradient in m/m
%.................................................
PROCEDURE press_grad: real
```

calib_press_trans

```
%.................................................
% Procedure  :  calib_press_trans
% Purpose    :  To enable user to calibrate a pressure
%               transducer
% Method     :  The user is prompted apply three pressure
%               differences to the transducer, as well
%               as to do one measurement with open shunt
%               valve (to determine voltage at zero pressure
%               difference). A second-order polynomial is
%               fitted to the difference measurements
%               and the coefficients are corrected for
%               the open shunt voltage
% Note      :   Calibration coefficients are based on
%               pressure in Pa!
%.................................................
PROCEDURE calib_press_trans(transd: nat1)
```

3.2.5 FLOWLIB

```
%.................................................
% Library  : flowlib
% Purpose  : To provide procedures that have to do with the
%            measurement with parameters of the process, like
%            temperature and flow rate.
% Required libraries:
%            gsd:iolib.gsc
%            hwd:st7.hwc
%            gsd:adclib.gsc
%.................................................
```

init_fbwtemp

```
%.................................................
% Procedure: init_flowtemp
% Purpose  : To get a first guess for values of fw_end, t_end
%            dp_vao_end (or to reset the values to bypass
%            moving average)
% Revision 28-05-96: arrays a_dp etc. filled in one shot
%.................................................
PROCEDURE init_flowtemp
```

fw_end

```
%.................................................
% Procedure: fw_end
% Purpose : to return the average flow_rate (computed from a_flow)
%.................................................
PROCEDURE fw_end: real
```

t_end

```
%.................................................
% Procedure: t_end
% Purpose : to return the average temperature (from a_temp)
%.................................................
PROCEDURE t_end: real
```

dp_vao_end

```
%.................................................
% Procedure: dp_vao_end
% Purpose : to return the average pressure drop voltage (from a_dp)
%.................................................
PROCEDURE dp_vao_end: real
```

fbwtemp

```
%.................................................
% Process  : flowtemp
% Purpose  : To continuously measure temperature, flow and pressure
%            difference
% Interface: This process modifies the global variables t_end, fw_en
%            and dp_end
% Remarks  : The process keeps some kind of running mean
%            The sampling interferes with the control of the steppen
%            motors, so abort flowtemp before moving steppenmotors.
% Revision 28-05-96: now data are stored in an array (a_dep etc.)
%            and average over array is computed when needed in
%            seperate routines (dp_end etc.)
%.................................................
PROCESS flowtemp[5000]
```

show_reynolds

```
%.................................................
% Procedure: show_reynolds
% Purpose  : writes Reynolds number, flow and temperature to
%            screen, updates every five seconds until a key is
%            pressed.
% Interface : x        IN        x coordinate of upper left corner
%                                of box where data are displayed.
%            y        IN        y coordinate of upper left ....
%            t        IN        time interval after which data
%                               should be updated (sec.)
%.................................................
PROCESS show_reynolds[10000]
```

temp

```
%.................................................
% Procedure: temp
% Purpose  : To get one sample of water temperature
% Interface: return value     temperature
%            temp shares signal s with flow;
% Remarks  : Temperature is measured via ADC adc_flow, channel 1
%            Measurement is externally triggered
%.................................................
PROCEDURE temp: nat2
```

fbw

```
%.................................................
% Procedure: flow
% Purpose  : To return one sample of the flowmeter
% Interface: return value    flow
% Q=10*N/4095/R * Qmax/Imax  met R=471 Ohm, Qmax = 78, Imax=20mA
% flow wordt gemeten via ADC adc_flow, kanaal 1
%.................................................
PROCEDURE flow: nat2
```

start_show_r, stop_show_r

```
%.................................................
% Procedure   :  start_show_r
% Purpose     :  To start display of Reynolds number and
%                other data on screen.
```

```
%................................................
PROCEDURE start_show_r
%................................................
% Procedure   :  stop_show_r
% Purpose     :  To stop display of Reynolds number and
%                other data on screen.
%................................................
PROCEDURE stop_show_r
```

3.2.6 IOLIB

```
%................................................
% Library   :  iolib
% Purpose   :  This library contains IO functions
% Author    :  Some functions are based on work by Roger van
%              Galen. Other procedures have been
%              written by Arnold Moene
% Date      :  Fall 1994
% Required libraries:
%              HWD:st5.HWC
% Note      :  Library only tested for PEP030 2.30.0
%................................................
```

write_real_arr

```
%................................................
% Procedure: write_real_arr
% Purpose  : To write a real value, formatted by writeln, to
%            a character array. At an end of line, a carriage-
%            line feed is added
% Interface: r         IN real value
%            arr      IN/OUT array to write to
%            pos      IN/OUT next position in array
%            eol      IN     is this end of line ?
%................................................
  PROCEDURE write_real_arr(r: real, arr: ARRAY 0 OF char,
      pos: REF integer, eol: boolean)
```

strcpy

```
%................................................
% Procedure  :  strcpy
% Purpose    :  PEP implementation of C-function strcpy
% Interface  :  strin    IN    string to be copied from
%               strout   OUT   string to be copied to
%................................................
PROCEDURE strcpy(strin: CONST string, strout: REF string)
```

beep

```
%................................................
% Procedure  :  beep
% Purpose    :  To ring the terminal bell (or beep)
%................................................
PROCEDURE beep
```

put_text

```
%................................................
% Procedure  :  put_text
% Purpose    :  writes text to standard output, starting
%               at given screen location
% Interface  :  x    IN    x-position on screen to start
%               y    IN    y-position on screen to start
%               text IN    string to put on screen
%................................................
PROCEDURE put_text(x, y: nat2, text: CONST string)
```

strcat

```
%................................................
% Procedure  :  strcat
% Purpose    :  a PEP implementation of the C-function strcat
% Note       :  Difference is that in this case the resulting
%               string is returned as a third parameter
%               in the parameter list, whereas in C the second
%               string is placed AT the END OF the first
%               string.
%               In PEP array extension is not possible.
% Interface  :  s1    IN    first string
%               s2    IN    second string
%               text  OUT   result string
%................................................
PROCEDURE strcat(s1, s2: CONST string,
                 text: REF string)
```

presskey

```
%................................................
% Procedure: presskey
% Purpose  : This is an overdeclaration of presskey in
%            HWD:ST5.HWC because OF some problems. Difference is
%            the position of the message
%................................................
PROCEDURE presskey
```

inq_str

```
%................................................
% Procedure : inq_str
% Purpose   : asks string of maximum length from standard input
%             (length = len) with question given by text
%             x_ and y_ are the screen coordinates where the
%             question starts.
%             A boolean is used to indicate whether string is
%             filled with at least one non-space character.
% Interface : x_, y_    IN    screen position of message
%             len       IN    maximum length of string
%             text      IN    message to display
%             str       OUT   string to be filled
%             filled    OUT   str has non-zero contents?
%................................................
PROCEDURE inq_str(x_, y_, len: nat2, text: CONST string,
                  str: REF string, filled: REF boolean)
```

inq_int, inq_real, inq_nat1, inq_bool

```
%................................................
% Procedures: inq_int, inq_real, inq_nat1, inq_bool
% Purpose   : prints question on standard output (text) and
%             reads the variable from standard input (answer).
%             Text is printed at screen coordinates x_, y_
% Interface : x_, y_    IN    screen coordinates of text
%             text      IN    text to display
%             answer    OUT   the variable read
%................................................
PROCEDURE inq_int(x_, y_: nat2, text: CONST ARRAY 0 OF char,
                  answer: REF integer)
PROCEDURE inq_real(x_, y_: nat2, text: CONST ARRAY 0 OF char,
                   answer: REF real)
PROCEDURE inq_nat1(x_, y_: nat2, text: CONST ARRAY 0 OF char,
                   answer: REF nat1)
PROCEDURE inq_bool(x_, y_: nat2, text: CONST ARRAY 0 OF char,
                   answer: REF boolean)
```

clear_presskey

```
%................................................
% Procedure  : clear_presskey
% Purpose    : removes line 'hit any key..' written by procedure
%              presskey defined above in this file
%................................................
PROCEDURE clear_presskey
```

make_title

```
%................................................
% Procedure  : make_title
% Purpose    : writes a menu title to screen, adding an
%              indication of the level of the current menu
%              (relative to main menu).
%              The procedure title can be found in HWD:st5.HWC
% Interface  : lev    IN    menu level
%              text   IN    menu name
%................................................
PROCEDURE make_title(lev: nat1, text: CONST string)
```

write_date

```
%................................................
% Procedure  :  write_date
% Purpose    :  writes a date passed as YYYYMMD in
%               DD-MM-YY format to standard output
%................................................
PROCEDURE write_date(dat: integer)
```

clear_line

```
%...................................................
% Procedure  :  clear_line
% Purpose    :  goes to specified line on screen and
%               overwrites existing text with spaces
% Interface  :  n    IN    line number to clear
%...................................................
PROCEDURE clear_line(n: nat1)
```

readbool

```
%...................................................
% Procedure :readbool
% Purpose   :To read in a file a 1 or 0 and sets a boolean
%            to true or false, respectively.
% Interface :f    IN    file to read from
%            b    OUT   boolean to read
%...................................................
PROCEDURE readbool(f: file, b: REF boolean)
```

writebool

```
%...................................................
% Procedure  :  writebool
% Purpose    :  writes a boolean to a file as 0 or 1
% Interface  :  f    OUT    file to write to
%               b    IN     boolean to write
%...................................................
PROCEDURE writebool(f: file, b: boolean)
```

ifwrite

```
%...................................................
% Procedure  :  ifwrite
% Purpose    :  To write a string to a file, only if a
%               boolean is true.
% Method     :  if boolean is true, write '1' to file,
%               followed by the string. Else only write '0'
%               The counterpart of this procedure is ifread
% Interface  :  b    IN        boolean
%               f    IN/OUT    file to write to
%               s    IN        string to write
%...................................................
PROCEDURE ifwrite(b: boolean, f: file, s: CONST string)
```

ifread

```
%...................................................
% Procedure  :  ifread
% Purpose    :  To conditionally read a string
% Method     :  a boolean is read from file (either 1 or 0),
%               If the boolean is true, the string is read.
%               The counterpart of this procedure is ifwrite
% Interface  :  b        OUT    boolean to read
%               f        IN     file to read from
%               s        OUT    string to read
%...................................................
PROCEDURE ifread(b: REF boolean, f: file, s: REF string)
```

check_dev

```
%...................................................
% Procedure  : check_dev
% Purpose    : checks whether a given device is a valid device
% Interface  : dev        IN    device name
%              vax_ldn    IN    VAX (0) ot LDN (1)
%...................................................
PROCEDURE check_dev(dev: CONST string,
                    vax_ldn: CONST nat2): boolean
```

check_fname

```
%...................................................
% Procedure  :  check_fname
% Purpose    :  To check whether this string is a valid
%               EPEP filename (i.e. with a valid
%               device name and valid extension).
% Interface  :  x, y    IN  screen coordinates for error
%                           messages (no message if x=y=0)
%               name    IN  proposed file name
%               vax_ldn IN  vax(0) OR ldn(1)
%...................................................
PROCEDURE check_fname(x, y: CONST nat2, name: REF string,
                      vax_ldn: CONST nat2): boolean
```

make_fname

```
%...................................................
% Procedure  :  make_fname
% Purpose    :  create a valid PEP file name from the
%               user ID, the file type (a character)
%               and the actual name
% Interface  :  uid    IN    user ID
%               name   IN    file name (between : and .)
%               type    IN    last character of extension
%               device  IN    device name
%               vax_ldn IN    VAX (0) or LDN (1)
%               fname   OUT   constructed name
%...................................................
PROCEDURE make_fname(uid: CONST ARRAY 2 OF char,
                     name: CONST ARRAY 8 OF char,
                     type: CONST char,
                     device: CONST ARRAY 3 OF char,
                     vax_ldn: CONST nat2,
                     fname: REF ARRAY 16 OF char)
```

inq_fname

```
%...................................................
% Procedure  :  inq_fname
% Purpose    :  To ask for a file name and check whether
%               it is a valid EPEP file name.
% Interface  :  x, y    IN    coordinates for question
%               text    IN    question
%               name    OUT   file name
%               valid   OUT   a valid name ?
%               vax_ldn IN    VAX (0) or LDN (1)
%...................................................
PROCEDURE inq_fname(x, y: nat2, text: CONST string,
                    name: REF string, valid: REF boolean,
                    vax_ldn: CONST nat2)
```

fill_name_array

```
%...................................................
% Procedure : fill_name_array
% Pupose    : To fill an element in an array of strings
%             This is a way to circumvent the fact that
%             assignments to strings do not work.
% Interface : t           IN   string to write to array
%             curr        IN   place in array to write
%             arr         IN/OUT  array to write to
%...................................................
PROCEDURE fill_name_array(t: CONST string,
                          curr: integer,
                          arr: ARRAY 0, 0 OF char)
```

3.2.7 LDALIB

```
%...................................................
% Library  :  ldalib
% Purpose  :  Contains procedure related to LDA
%             measurements and flow / temperature
%             measurements..
% Author   :  Wiendelt Steenbergen / Arnold Moene
% Date     :  1993-1995
% Notes    :  diode 1 = component U + V
%             diode 2 = component U - V
%             (with V vertically upward)
% Required libraries:
%             HWD:st7.HWC
%             GSD:iolib.GSC
%             GSD:ldadec.GSC
%             GSD:parslib.GSC
%             GSD:adclib.GSC
% Required compiled code:
%             GSD:sommatie.GSO
%             GSD:trackcor.GSO
%             GSD:checkloc.GSO
% Note     :  Library tested only for PEP030 2.30.0
%...................................................
```

sommatie

```
%...................................................
% Procedure: sommatie
% Purpose  : To calculate averages, variances and covariances
```

```
%            of two signals (with value between 0-4095)
% Interface: av            OUT    averages of a and b
%            mom20 mom21   OUT    2nd moments
%            mom300 mom301 OUT    3rd moments
%            mom310 mom311 OUT    3rd moments
%            underflow     OUT    number of samples with
%                                 underlow
%            overflow      OUT    number of samples with
%                                 overflow
%            dat_a dat_b   IN     arrays with ADC data
%            psample       IN     number of samples
%            do3           IN     should I calculate 3rd
%                                 moments?
% Remark   : Each datapoint is checked for overflow and
%            underflow and the number of accurences is
%            exported the use of one-dimensional arrays for
%            mom2 and mom3 is due to the incapability OF the
%            pep-compiler to deal properly with
%            multi-dimensional arrays.
%.................................................
PROCEDURE sommatie(av: ARRAY 0 OF real,
   mom20: ARRAY 0 OF real,
   mom21: ARRAY 0 OF real,
   mom300: ARRAY 0 OF real,
   mom310: ARRAY 0 OF real,
   mom301: ARRAY 0 OF real,
   mom311: ARRAY 0 OF real,
   underflow, overflow: REF integer,
   meet_a, meet_b: ARRAY 0 OF nat2, psample: integer,
   do3: boolean) AT sommatie_code[0]
```

check_lock

```
%.................................................
% Procedure: check_lock
% Purpose  : to  remove data from two velocity arrays, for
%            which not both trackers did lock
% Interface: u_dat, v_dat IN/OUT  nat2 data to be checked
%            u_loc, v_loc IN      lock data
%            lock_limit   IN   count above which locking is on
%            new_samp     IN/OUT  number of samples
%.................................................
PROCEDURE check_lock(udat, vdat, uloc, vloc: ARRAY 0 OF nat2,
    lock_limit: nat2, new_samp: REF integer) AT check_code[0]
```

track_corr

```
%.................................................
% Procedure: track_corr
% Purpose  : To apply calibration curve to tracker data
% Interface: uu     IN/OUT   array with ADC data from u
%                            tracker
%            vv     IN/OUT   array with ADC data from v
%                            tracker
%            sample IN       number of samples in arrays
%            ampmp  IN       multiplier of amplifiers
%            ampos  IN       offset of amplifiers
%            parmp  IN       multiplier of PARSAM
%            nul_c  IN       count value at 0 Volt
%            vper_c IN       number of counts per volt
%            r1     IN       range of tracker 1
%            r2     IN       range of tracker 2
%            cof    IN       calibration coefficients
%.................................................
PROCEDURE track_corr(uu, vv: ARRAY 0 OF nat2, sampl: integer,
    ampmp: ARRAY 0 OF real, ampos: ARRAY 0 OF real,
    parmp: ARRAY 0 OF real, vper_c: real,
    ri1, ri2: nat1, cof: ARRAY 0 OF real, nul_c: nat2)
AT corr_code[0]
```

moments

```
%.................................................
% Procedure: moments
% Purpose  : conversion OF moments(1 st, 2 nd AND 3 rd) from
%            adc counts in lda frame to real world velocities
%            and dir
% Interface:  Global variables used: scf[], mp[], os[], f_s[],
%             def_range[] and rangeindex[]
%             av_in    IN        averages in LDA frame
%             mom2_in  IN        2nd moments in LDA frame
%             mom3_in  IN        3rd moments in LDA frame
%             av       IN/OUT    averages
%             mom2     IN/OUT    2nd moments
%             mom3     IN/OUT    3rd moments
% Remark   :  su, sv etc. are assumed to be produced by
%             sommatie .
%             That procedure produces the actual moments av(u),
%             av(v), av(u'u'), av(v'v'), av(u'v').
% Author   :  W. Steenbergen, A.F. Moene
% Data     :  08-07-1994 (rev. 13-7-94)
%             03-10-1995: adjustment to use of mp, os, p_gain
%             21-11-1995: adjustment in computation of aaa
%.................................................
PROCEDURE moments(av_in: ARRAY 0 OF real,
    mom2_in: ARRAY 0, 0 OF real,
    mom3_in: ARRAY 0, 0, 0 OF real,
    av: REF ARRAY 0 OF real,
    mom2: REF ARRAY 0, 0 OF real,
    mom3: REF ARRAY 0, 0, 0 OF real)
```

density

```
%.................................................
% Procedure  : density
% Purpose    : To compute density of water as function of
%              temperature
% Interface  : temperat    IN  temperature in K
%              return value    density in kg/m3
%.................................................
PROCEDURE density(temperat: real): real
```

viscos

```
%.................................................
% Procedure  : viscos
% Purpose    : To compute the kinematic viscoity (as
%              function of temperature
% Interface  : temperat    IN  temperature in K
%              return value    viscosity in m**2/s
%.................................................
PROCEDURE viscos(temperat: real): real
```

reynolds

```
%.................................................
% Procedure  :  reynolds
% Purpose    :  To calculate Reynolds number from flow rate
%               and temperature.
% Interface  :  t    IN    temperature in deg. Celsius
%               fl   IN    flowe rate in m3 / hour
%               rad  IN    radius of pipe in m
%.................................................
PROCEDURE reynolds(t, fl, rad: real): real
```

calib_track

```
%.................................................
% Procedure  :  calib_track
% Purpose    :  To enable user to calibrate a tracker (i.e.
%               fill a part of the calibration table)
% Method     :  The user is prompted to set the signal
%               generator to a frequency. Is this has been
%               done, samples are collected and the
%               deviation of the voltage from the ideal
%              inear calibration is determined. This
%               procedure is repeated 3 times. Then
%               a second-order polynome is fitted through
%               the deviations for three frequencies.
% Interface  :  range      IN   the freq. ranges of each of the
%                               two trackers to be calibrated
%.................................................
PROCEDURE calib_track(range: ARRAY 1..2 OF nat1)
```

3.2.8 MATHLIB

```
%.................................................
% Library  :  mathlib
% Purpose  :  Contains mathematical procedures
% Author   :  Arnold Moene
% Date     :  fall 1994
% Revision :  october 1995 : addition of deg2rad and rad2deg
% Required libraries:
%             none
%.................................................
```

deg2rad

```
%.................................................
% Procedure: deg2rad
```

```
% Purpose  : To convert an angle in degrees to radians
% Interface: r           in     angle in degrees
%            return value       angle in radians
%.................................................
PROCEDURE deg2rad(r: real): real
```

rad2deg

```
%.................................................
% Procedure: rad2deg
% Purpose  : To convert an angle in radians to degrees
% Interface: r            in   angle in radians
%            return value      angle in degrees
%.................................................
PROCEDURE rad2deg(r: real): real
```

fbor

```
%.................................................
% Procedure: floor
% Purpose  : To return largest integer that is smaller than a
%            given real value
% Interface: r        in        real value
%            return value       the requested integer value
%.................................................
PROCEDURE floor(r: real): integer
```

trunc

```
%.................................................
% Procedure: trunc
% Purpose  : To return largest integer that is smaller than a
%            given real value
% Interface: r        in        real value
%            return value       the requested integer value
% Remark   : This procedure exists for compatibility reasons.
%            Some libraries use trunc instead of floor.
%.................................................
```

ceil

```
%.................................................
% Procedure: ceil
% Purpose  : To return the smallest integer value that is larger
%            than a given real value
% Interface: r        IN        real value
%            return value       requested integer value
%.................................................
PROCEDURE ceil(r: real): integer
```

pow

```
%.................................................
% Procedure: pow
% Purpose  : To return x**y, where x should be a positive
%            integer and y should be an integer.
% Interface: x        IN        base
%            y        IN        exponent
%            return value       x**y (real)
% Remark   : This procedure is needed since PEP does not
%            allow (though it should) negative exponents.
%.................................................
PROCEDURE pow(x, y: integer): real
```

determinant3

```
%.................................................
% Procedure: determinant3
% Purpose  : to determine determinant of 3x3 matrix
% Interface: ccc     IN    3x3 matrix
%            return value  determinant of ccc
%.................................................
PROCEDURE determinant3(ccc: REF ARRAY 1..3, 1..3 OF real): real
```

determinant2

```
%.................................................
% Procedure: determinant2
% Purpose  : To determine determinant of 2x2 matrix
% Interface: matr    IN    2x2 matrix
%            return value  determinant of matrix
%.................................................
PROCEDURE determinant2(matr: REF ARRAY 1..2, 1..2 OF real): real
```

absr

```
%.................................................
% Procedure: absr
% Purpose  : To return the absolute value of a real
% Interface: r     IN      real of which to determine absolute
%                          value
%            return value  absolute value
%.................................................
PROCEDURE absr(r: real): real
```

3.2.9 PARSLIB

```
%.................................................
% Library   :  parslib
% Purpose   :  Procedures to control ParSam
% Author    :  Most of this library is due to Roger van
%              Galen. Some additions by Arnold Moene.
% Date      :  Fall 1994
% Required libraries:
%              none
% Required compiled code:
%              GSD:split.GSO
% Note      :  Library only tested for PEP030 2.30.0
%.................................................
```

split

```
%.................................................
% Procedure  :  split
% Purpose    :  To split pairs of samples, stored as
%               integers, into 2 2-byte naturals.
% Note       :  This is a compiled procedure.
% Interface  :  g_    IN    array with integers
%               pd_   OUT   array of 2-byte nats
%               n_    IN    number of samples
%.................................................
PROCEDURE split(g_: ARRAY 0 OF integer, pd_: ARRAY 0 OF nat2,
               n_: integer) AT parcomcode[0]
```

pars_gain_all

```
%.................................................
% Procedure  :  pars_gain_all
% Purpose    :  To set gains of all channels to same value
% Method     :  The gain is set to 2**v_
% Interface  :  v_    IN    power of 2 to set gains to
%.................................................
PROCEDURE pars_gain_all(v_: integer)
```

pars_gain

```
%.................................................
% Procedure  :  pars_gain
% Purpose    :  To set gain of specified channel to
%               specified value.
% Method     :  Gain is set to 2**v_.
% Interface  :  nr_    IN    channel number
%               v_     IN    power of 2 to set gain to
%.................................................
PROCEDURE pars_gain(nr_: nat2 v_: integer)
```

n_pars_gain

```
%.................................................
% Procedure  :  n_pars_gain
% Purpose    :  To set gain of specified range of
%               channels to specified value.
% Method     :  Gain is set to 2**v_.
% Interface  :  k1_    IN    first channel number
%               k2_    IN    last channel number
%               v_     IN    power of 2 to set gain to
%.................................................
PROCEDURE n_pars_gain(k1_, k2_: nat2 v_: integer)
```

w_pars_gain

```
%.................................................
% Procedure  :  w_pars_gain
% Purpose    :  To read current gain of specific channel
```

```
% Interface  :  nr_    IN    channel number
%               RETURN       gain of channel nr_
%.................................................
PROCEDURE w_pars_gain(nr_: integer): integer
```

ena

```
%.................................................
% Procedure  :  ena
% Purpose    :  To start sampling by PARSAM
%.................................................
PROCEDURE ena
```

disa

```
%.................................................
% Procedure  :  disa
% Purpose    :  To stop sampling by PARSAM
%.................................................
PROCEDURE disa
```

p_status

```
%.................................................
% Procedure  :  p_status
% Purpose    :  To write contents of certain Phybys
%               registers used by the PARSAM (see
%               (see PARSAM manual p. 11 ->)
% Note       :  register contents is written in binary
%               form as well as in decimal form
% Interface  :  n    IN    number of register
%                          (relative to PARSAM base address)
%.................................................
PROCEDURE p_status(n: integer)
```

clearmem

```
%.................................................
% Procedure  :  clearmem
% Purpose    :  To clear specified memory (i.e. give free
%               to overwrite)
% Interface  :  n    IN    number of memory (0 or 1)
%.................................................
PROCEDURE clearmem(n: integer)
```

n_chan

```
%.................................................
% Procedure  :  n_chan
% Purpose    :  To tell PARSAM how many channels have to be
%               used during sampling.
% Interface  :  n    IN     number of channels
%.................................................
PROCEDURE n_chan(n: integer)
```

init

```
%.................................................
% Procedure  :  init
% Purpose    :  To initialize PARSAM for use with a
%               specified number of channels
% Interface  :  n_    IN    number of channels
%.................................................
PROCEDURE init(n_: integer)
```

maxsamp

```
%.................................................
% Procedure  :  maxsamp
% Purpose    :  To set maximum number of samples to be
%               collected per channel
%.................................................
PROCEDURE maxsamp(i_: integer)
```

cro

```
%.................................................
% Procedure  :  cro
% Purpose    :  To set the number of the channel to be
%               read out next to a specific value.
% Interface  :  n_    IN    number of the channel of which
%                           data should be read next.
%.................................................
PROCEDURE cro(n_: integer)
```

sos

```
%.................................................
% Procedure  :  sos
% Purpose    :  To set the storage option select bits.Valid
%               values are:
%               0: offset binary ADC data (-10 to 9.9951 V =
%                   0 - 4095
%               1: 2-complements ADC data (-10 to 9.9951 V =
%                   -2048 - 2047
%               2: add channel number to ADC data to
%                  2-complements data (in bits 15-12)
%               3: same as 0
% Interface  :  i_    IN    storage option
%.................................................
PROCEDURE sos(i_: integer)
```

wacht

```
%.................................................
% Procedure  :  wacht
% Purpose    :  To wait until PARSAM has one memory filled
% Note       :   Value of maxwacht is set in initit
%.................................................
PROCEDURE wacht
```

initit

```
%.................................................
% Procedure  :  initit
% Purpose    :  To initialize variables used in this
%               library, and to initialize the PARSAM.
%.................................................
PROCEDURE initit
```

meet

```
%.................................................
% Procedure  :  meet
% Purpose    :  To check the time it takes to fill the
%               memories, given the current settings.
%               After a memory has been filled, the number
%               of the memory is given, followed by the
%               time it took to fill it (in ms)
% Note       :  This is a revised version of the
%               original by Roger van Galen
%.................................................
PROCEDURE meet
```

loopje

```
%.................................................
% Procedure :  loopje
% Purpose   :  To sample a given number of channels and
%              collect a given number of samples.
% Interface :  nchan    IN   number of channels
%              maxi     IN   number of samples
%.................................................
PROCEDURE loopje(nchan, maxi: integer)
```

botte_adcs

```
%.................................................
% Procedure  :  botte_adcs
% Purpose    :  To use the PARSAM as a straight ADC.
% Interface  :  n_    IN    number of channels to sample
%.................................................
PROCEDURE botte_adcs(n_: integer)
```

startparsam

```
%.................................................
% Procedure  :  startparsam
% Purpose    :  To initialize PARSAM for a given number of
%               channels and given number of samples, and
%               to start sampling.
% Note       :  Strorage option selection: offset binary data
%               This is a modified version of the startparsam
```

```
%               by Roger van Galen.
% Interface  :  n_    IN    number of channels
%               i_    IN    number of samples
%................................................
PROCEDURE startparsam(n_, i_: integer)
```

parsam_fill_array

```
%................................................
% Procedure  :  parsam_fill_array
% Purpose    :  To fill an array of given length with data
%               from a given number of channels, in a given
%               range.
% Interface  :  f_      IN    first channel number
%               l_      IN    last channel number
%               n_      IN    number of channels
%               pardat  IN    array with pairs of ADC
%                             samples for all channels
%               nsamp   IN    number of samples
%................................................
PROCEDURE parsam_fill_array(f_, l_, n_: CONST integer,
                            pardat: REF ARRAY 0, 0 OF integer,
    nsamp: CONST integer)
```

3.2.10 STEPLIB

```
%................................................
% Library  :  steplib
% Purpose  :  To provide routines for control of stepper
%             motors for LDA system.
% Author   :  Most routines have been written by Wiendelt
%             Steenbergen. Modifications and new routines
%             are due to Arnold Moene
% Date     :  1993-1995
% Required libraries:
%             HWD:st5.HWC
%             GSD:iolib.GSC
%             GSD:ldadec.GSC
%             GSD:ldalib.GSC
%................................................
```

do_step

```
%................................................
% Procedure: do_step
% Purpose  : To do a certain number of steps in some direction
%            with a cetrain stepper motor
% Interface: mo        IN        motor number
%            st        IN        number of steps
%            dir       IN        direction (1 or 0)
%................................................
PROCEDURE do_step(mo, st, dir: nat2)
```

get_count

```
%................................................
% Procedure: get_count
% Purpose  : To get the value of the position counter of one
%            of the two moving devices (vertical or horizontal)
% Interface: ch      IN    the number of the moving device
%                          (1 : horizontal, 2: vertical)
%................................................
PROCEDURE get_count(ch: integer): integer
```

set_count

```
%................................................
% Procedure: set_count
% Purpose  : To set the value of the position counter of
%            one of the two moving devices (vertical or
%            horizontal
% Interface: ch      IN    the number of the moving device
%                          (1 : horizontal, 2: vertical)
%            offs    IN    the value to set the counter to
%................................................
PROCEDURE set_count(ch, offs: integer)
```

move_y

```
%................................................
% Procedure: move_y
% Purpose  : to move the measuring volume to a given y-position
%            The position is given in steppenmotor steps, which
%            equal 0.02 mm
% Interface: pos          IN    position in steps
%................................................
PROCEDURE move_y(pos: integer)
```

move_z

```
%................................................
% Procedure: move_z
% Purpose  : to move the measuring volume to a given
%            z-position. The position is given in stepper
%            motor steps which equal 0.02 mm
% Interface: pos          IN    position in steps
%................................................
PROCEDURE move_z(pos: integer)
```

reset

```
%................................................
% Procedure: reset
% Purpose  : to move measuring volume to centre of pipe
% Interface: none
%................................................
PROCEDURE reset
```

to_rr

```
%................................................
% Procedure: to_rr
% Purpose  : moves the measuring volume to a position given in
%           r/R coordinates, and for a given traversing angle
% Interface: r_over_r        IN    position
%            trav_angle      IN    traversing angle
%................................................
PROCEDURE to_rr(r_over_r, trav_ang: CONST real)
```

switch_on

```
%................................................
% Procedure  :  switch_on
% Purpose    :  To keep LDA at same position after
%               switching on of stepper motors.
% Method     :  In this respect, the values returned by the
%               encoders give the absolute position. The
%               values before and after switching on of the
%               stepper motors are compared. If they
%               differ more than one step of either of the
%               stepper motors, the position is corrected.
%  Interfac  :  x_    IN    x-position on screen of message
%               y_    IN    y-position on screen of message
%................................................
PROCEDURE switch_on(x_, y_: nat2)
```

A File formats

A.1 File format for processed data

The processed LDA data are written to an ASCII file. However, this file is a binary file due to the limitations in PEP on the number of characters on a line. Each line is written as an array of characters, with a carriage-return/line feed as the last character. This arrangement implies some postprocessing before the file can be read as a normal text file [1]. After the data of a given radial position have been processed, the data of *all* radial positions, including the last, will be written to file. Data of one radial position occupy one line in the output file (see table A.1.

Here u is the velocity component in the axial direction and v is the velocity component prependicular to both u and the plane through the two reference beams. T is the water temperature, Q is the flow rate, Re is the Reynolds number (based on bulk velocity and pipe diameter), $\partial p/\partial z$ is the axial pressure gradient (if measured). u_* is the friction velocity and y^+ is the distance from the wall in wall units. The latter three quantities only have non-zero values when the pressure gradient is measured (this can be done simultaneously with LDA measurements). N_{samp} is the number of samples used in the calculation of the velocity statistics.

In the case that only pressure data are stored the format of the files is as shown in table A.2.

Here x_1 and x_2 are the two axial positions of the pressure taps used for the measurements. The last column in the file (transducer) indicates whether the the DP15 (1) or the DP45 (2) transducer has been used.

A.2 File format of binary file for raw data

The raw data are stored in a binary file for reasons of speed and disk space. In order to make the file portable, all information is written in 2-byte integers (`unsigned short` in C, `nat2` in PEP) and 1-byte integers (`char` in C, `character` in PEP).

An integer *n* is stored as two `nat2` variables:

```
nat2[0] := n + 2**15 DIV 2**16
nat2[1] := n + 2**15 MOD 2**16
```

A real *r* is stored as two `nat2` variables as follows:

```
mant = ((float) mant\_exp[0]) - 32768
expo = ((float) mant\_exp[1]) - 32768
r = mant * pow(2.0, expo)
```

The file starts with a header, containing the data show in table A.3.

Some notes are in place here. The experiment code occupies 8 bytes, corresponding to the space of 4 `nat2` variables. For the `datatype`, the following codes are valid (they are defined in `ldadec.gsc`) which are given in table A.4.

[1] On UNIX a solution is to pass `inputfile` through the following pipe to obtain `outputfile`: `cat inputfile|sed ''| awk '{for (i=1;i<=NF;i++) {printf("%s ", $i)}; if (NF > 0) {printf("\ n")}}' > outputfile`

column	quantity	unit
1	r/R	-
2	$\overline{u}$	m/s
3	$\overline{v}$	m/s
4	$\overline{u'u'}$	$(m/s)^2$
5	$\overline{v'v'}$	$(m/s)^2$
6	$\overline{u'v'}$	$(m/s)^2$
7	T	^oC
8	Q	m^3/hr
9	Re	-
10	$\partial p/\partial z$	mH_2O/m
11	u_*	m/s
12	y^+	-
13	N_{samp}	-

Table A.1: Format of processed data file.

column	quantity	unit
1	x_1	m
2	x_2	m
3	Δp	mH_2O
4	Q	m^3/hr
5	T	^oC
6	Re	-
7	$\partial p/\partial z$	mH_2O/m
8	transducer	1 or 2

Table A.2: Format of processed data file for pressure measurements.

`name`	type	#	description
`version`	nat2	1	file version number
`expcode`	character	8	experiment code
`datatype`	nat2	1	info on block type etc.
`ngroups`	nat2	1	number of radial positions in file

Table A.3: Format of file header of binary file.

		meaning			
PEP name	value	ADC	continuous	# blocks	lock data
old_adc_lock	0	old	yes	1	yes
con_1_lock	1	PARSAM	yes	1	yes
con_n_lock	2	PARSAM	yes	more	yes
nocon_n_lock	3	PARSAM	no	more	yes
old_adc_nolock	50	old	yes	1	no
con_1_nolock	51	PARSAM	yes	1	no
con_n_nolock	52	PARSAM	yes	more	no
nocon_n_nolock	53	PARSAM	no	more	yes

Table A.4: Codes for type of stored data.

The number of radial positions in the file `ngroups` is determined at the start of the measurements. It may be possible that at the measurements have not been made for *all* radial positions, and thus `ngroups` may be inaccurate.

The data for each radial position is described by a so-called group descriptor. This sequence is written just before the data of a given radial position (thus all group descriptors are scattered through the file). If `datatype` equals `con_n_nolock`, `con_n_nolock`, `nocon_n_lock` or `nocon_n_nolock`, the group descriptor starts with the integer (PEP-type `nat2`) named `totcnt`, which is the number of blocks in which the data has been written. In the other cases `totcnt` equals one by definition. The rest of the group descriptor contains the data given in table A.5.

Then it is finally time to actually read the data. If the storage type is a multi-block type (`con_n_nolock`, `con_n_nolock`, `nocon_n_lock` or `nocon_n_nolock`) the data are preceded by the number of the block (stored as a `nat2`) and the number of samples in the block (an `int`, stored as two `nat2` variables). Then the (blocks of) actual data are stored: for the storage types without lock-information these are two arrays of integer data (integers, since that is the way it had been stored in the PARSAM memory). For the storage types with lock information four arrays of data are present.

The integer data arrays have to be split since what was stored and read as an array of integers in fact contained data of type `nat2` (or unsigned shorts in C). Teh data are split as follows (from a C program):

```
/* (3.1) Get two samples */
samp0 = (unsigned short)
        (intdat[i] >> 16) ;
samp1 = (unsigned short)
        (intdat[i] - ((unsigned int) samp0 << 16));

/* (3.2) Exchange samples and swap bytes in sample */
nat2dat[2*i+1] = (unsigned short) (samp0 >> 8) +
                 (unsigned short) (samp0 << 8);
nat2dat[2*i] = (unsigned short) (samp1 >> 8) +
               (unsigned short) (samp1 << 8);
```

name	type	#	description
r_r	real	1	radial position r/R
ustar	real	1	friction velocity u_* (m/s) (if pressure gradient measured as well)
re	real	1	Reynolds number Re
t	real	1	water temperature ^oC
q	real	1	flow rate (m^3/hr)
yplus	real	1	distance from the wall in plus units
nsamp	int	1	total number of samples for this point
freq	int	1	sampling frequency (Hz)
cal	real	6	calibration factors for the 2 trackers (elements 0,1,2 for tracker 1)
conv	real	4	2×2 matrix to convert velocities from phyisical frame to LDA-frame
travang	real	1	traversing angle (radians)
ldaang	real	1	angle of LDA (radians)
mp	real	2	gains of amplifiers
pgain	real	2	gains of PARSAM
os	real	2	offsets of amplifiers
range	int	2	range of tracker (between 1 and 7)
shift	int	2	shift frequency (Hz)

Table A.5: Data in group header in binary file.

Bibliography

Smeets, R. and Nijmweegen, F. v. (1993). *User manual PhyBUS PARSAM-25/250 Parallel Sampling ADC*. Physical and Technical Laboratory Automation Group; Department of Physics; University of Technology Eindhoven, Einhoven, first edition. BL 92-02.

Steenbergen, W. (1995). *Turbulent pipe flow with swirl*. Ph.D. thesis, Eindhoven University of Technology.

Voskamp, J. H., Nijmwegen van, F. C., and Wal van der, A. J. (1989). PHYDAS, een integraal systeem voor data-acquisitie en besturing van fysische experimenten (*in dutch*). *Ned. Tijdschr. v. Natuurk.*, **A55**(1/2), 50–55.